Escuela Politécnica Superior

Creación de un videojuego de terror usando Unreal Engine 4

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Sergio Molero García

Tutor/es:
Mireia Luisa Sempere Tortosa

Universitat d'Alacant
Universidad de Alicante

Septiembre 2018

Agradecimientos

En estos años he pasado por numerosas experiencias, que me han hecho crecer como persona y han despertado mi pasión por el mundo 3D.

En especial, darles las gracias a mis padres, por estar siempre apoyándome y dándome ánimos para terminar este proyecto, y los siguientes que vengan.

También muchas gracias a mis amigos y compañeros de trabajo, los cuales me han dado ratos de diversión, maravillosos días de clase y anécdotas que nunca olvidaré. En especial agradecerles a Manu y Fran, por siempre estar ahí en los momentos más difíciles y aguantarme.

Con este trabajo doy fin a cuatro años que no cambiaría por nada del mundo, en los que he conocido a gente maravillosa que comparte mis gustos.

Resumen

En el mundo de los videojuegos, hay diferentes roles, como el de programador, diseñador y artista 3D, entre muchos otros. Como artista 3D que quiero ser, quiero realizar un trabajo de modelado, pero sin dejar de lado el resto de los roles dentro de un equipo de desarrollo de videojuegos.

El objetivo de este trabajo de fin de grado es realizar un videojuego de terror utilizando el motor gráfico *Unreal Engine* 4 y usando sólo elementos realizados desde cero, diseñados por mí. Se realizará el documento de diseño de un videojuego o GDD con toda la información y el proceso de creación de este.

También se realizará un estudio sobre la actualidad del género de terror, diferentes motores gráficos, programas de modelado y texturizado, todo ello acompañado además de fotografías del proceso de creación del nivel del videojuego.

El resultado final, ***Psycho***, busca ser una experiencia completa, tanto visual como sonora, en la que el jugador explore la casa, resuelva los misterios y descubra la historia de esta.

Contenido

Índice de figuras 12
1.Introducción 17
2. Marco teórico o Estado del arte 18
2.2¿Qué es un videojuego? 18
2.2. Referencias 20
2.2.1. *Emily Wants To Play* 21
2.2.2. *Play With Me* 22
2.2.3. P.T 23
2.2.4. *Layers of fear* 24
2.2.5. *Resident Evil* 7 26
3. Objetivos 28
3.1. Desglose de objetivos 28
4. Metodología 29
4.1. Kanban 29
4.2. Gestión del proyecto 30
4.3. Control de versiones 30
5. Cuerpo del trabajo 31
5.1. Elección de herramientas 32
5.1.1. Motores Gráficos 32
5.1.1.1. *Unity* 32
5.1.1.2. *Unreal Engine* 33
5.1.1.1. Elección del motor 34
5.1.2. Software de Modelado 35
5.1.2.1. *Blender* 35
5.1.2.2. *Maya* 36
5.1.2.3. 3ds Max 37
5.1.3. Texturizado 38
5.1.3.1. Mudbox 38
5.1.3.2. Substance Painter 38
5.1.3.3. *Photoshop* 39
5.1.4. Sonido 40
5.1.4.1. *FreeSound* 40

5.1.4.2. *Audacity* 40
5.2. Documento de Diseño del Videojuego (GDD) 41
5.2.1. Conceptos del juego 41
5.2.1.1. Argumento 41
5.2.1.2. Objetivo del juego 42
5.2.1.3. Conjunto de características 42
5.2.1.4. Género 43
5.2.1.5. Plataforma 43
5.2.1.6. Tecnología 43
5.2.1.7. Público 43
5.2.2. Jugabilidad 44
5.2.3. Mecánicas 45
5.2.3.1. Movimiento y cámara 46
5.2.3.2. Interacción 47
5.2.3.3. Inspección 48
5.2.3.4. Enfoque 49
5.2.3.5. Linterna 50
5.2.3.6. Piezas Secretas 51
5.2.4. Personajes 52
5.2.5. Nivel del juego 53
5.2.6. Progresión del Juego 54
5.3. Desarrollo 55
5.3.1. *Unreal Engine* 55
5.3.1.1. *Blueprints* 57
5.3.1.1.1. ¿Cómo funcionan? 57
5.3.1.1.2. Tipos de *blueprints* 58
5.3.1.1.2.1. *Blueprint* de nivel 58
5.3.1.1.2.2. *Blueprint* de clase 59
5.3.1.1.2. ¿Qué más pueden hacer los *blueprints*? 60
5.3.1.2. Luces 61
5.3.1.2.1. Tipos de luces 61
5.3.1.2.1. Propiedades de las luces 63
5.3.1.3. Materiales 64
5.3.1.2.1. Entradas 66

5.3.1.2.1.1. *Base Color* 66
5.3.1.2.1.2. *Metallic* 67
5.3.1.2.1.3. *Specular* 68
5.3.1.2.1.4. *Roughness* 69
5.3.1.2.1.5. *Emissive Color* 70
5.3.1.2.1.6. *Opacity* 71
5.3.1.2.1.7. *Opacity Mask* 72
5.3.1.2.1.8. *Normal* 73
5.3.1.2.1.9. *Subsurface Color* 74
5.3.1.2.1.10. *Ambient Oclusion* 75
5.3.1.2.1.11. *Refraction* 76
5.3.1.4. Texturas 77
5.3.2. Creación del nivel 78
5.3.2.1. Paredes 78
5.3.2.2. Colisiones 80
5.3.2.3. Modelos 3D 83
5.3.2.3.1. Dormitorio 1 84
5.3.2.3.2. Dormitorio 2 85
5.3.2.3.3. Aseo 86
5.3.2.3.4. Cocina 87
5.3.2.3.5. Comedor 88
5.3.2.3.6. Pasillo 89
5.3.2.3.7. Galería 90
5.3.2.3.8. Habitación secreta 91
5.3.2.3.9. Recuento final 91
5.3.2.4. Texturizado 92
5.3.2.5. Materiales 96
5.3.3. Desarrollo de mecánicas 98
5.3.3.1. Movimiento y cámara 98
5.3.3.2. Interacción 101
5.3.3.3. Inspección 104
5.3.3.4. Enfoque 106
5.3.3.5. Linterna 107
5.3.3.6. Piezas Secretas 108
5.3.3.7. Eventos 110

5.3.4. Multimedia 111
5.3.4.1. Sonido 111
5.3.4.2. Video 118
5.3.5. Menús 121
5.3.5.1. Menú Principal 122
5.3.5.2. Controles 124
5.3.5.3. Pantalla de carga 125
5.3.5.4. Configuración 126
5.3.5.5. Cinemática inicial 129
5.3.5.6. Créditos 130
5.3.6. Realidad Virtual 131
6. Conclusiones 132
7. Bibliografía 133
8. Anexo: Arte 135

Índice de figuras

Figura 1. Imagen in-game de Emily Wants To Play ____ *21*
Figura 2. Imagen in-game de Play With Me ____ *22*
Figura 3. Imagen in-game de P.T. ____ *23*
Figura 4. Imagen in-game de Layers of Fear. ____ *24*
Figura 5. Imagen in-game de Resident Evil 7 ____ *27*
Figura 6. Imagen explicativa de la metodología Kanban ____ *29*
Figura 7. Elección de los dos finales del juego ____ *42*
Figura 8. Captura InGame del dormitorio1 ____ *46*
Figura 9. Captura InGame que muestra la interacción con una puerta ____ *47*
Figura 10. Captura InGame que muestra la inspección de un rotulador ____ *48*
Figura 11. Captura InGame de la cocina sin enfoque ____ *49*
Figura 12. Captura InGame de la cocina con enfoque ____ *49*
Figura 13. Captura InGame del uso de la linterna en el pasillo ____ *50*
Figura 14. Captura InGame del uso de la linterna en la cocina ____ *50*
Figura 15. Pieza secreta resaltada al mirarla ____ *51*
Figura 16. Piezas secretas colocadas según se han cogido ____ *51*
Figura 17. Boceto original de la estructura de la casa ____ *53*
Figura 18. Interfaz de Unreal Engine ____ *56*
Figura 19. Ejemplo de un blueprint de nivel ____ *58*
Figura 20. Ejemplo de un blueprint de clase ____ *59*
Figura 21. Ejemplo de las opciones del blueprint carácter ____ *60*
Figura 22. Ejemplo de blueprint de interfaz o HUD ____ *60*
Figura 23. Luz direccional ____ *61*
Figura 24. Luz puntual ____ *61*
Figura 25. Luz de foco ____ *62*
Figura 26. Luz ambiental ____ *62*
Figura 27. Tipos de movilidad de una luz ____ *63*
Figura 28.Material que representa un suelo de madera ____ *64*
Figura 29. Material que representa el agua de un océano ____ *65*
Figura 30. Ejemplo de entrada base color ____ *66*
Figura 31. Ejemplo de entrada metallic ____ *67*
Figura 32. Ejemplo de entrada specular ____ *68*
Figura 33. Ejemplo de entrada roughness ____ *69*
Figura 34. Ejemplo de entrada emissive color ____ *70*
Figura 35. Ejemplo de entrada opacity ____ *71*
Figura 36. Ejemplo de entrada opacity mask ____ *72*
Figura 37. Ejemplo de entrada normal ____ *73*
Figura 38. Ejemplo de entrada subsurface color ____ *74*
Figura 39. Ejemplo de entrada ambient oclusion ____ *75*
Figura 40. Ejemplo de entrada refraction ____ *76*
Figura 41. Pared de la casa texturizada en Substance Painter ____ *78*
Figura 42. Paredes de la casa en 3ds max con su correspondiente unwrap ____ *79*

Figura 43. Cimientos de la casa construidos ____ *79*
Figura 44. Altavoz con custom collision ____ *80*
Figura 45. Modelo del sofá del comedor ____ *81*
Figura 46. Creación de colisiones customizadas para el sofá ____ *81*
Figura 47. Nomenclatura para la correcta detección de la colisión personalizada ____ *82*
Figura 48. Modelos 3d del dormitorio 1 ____ *84*
Figura 49. Modelos 3d del dormitorio 2 ____ *85*
Figura 50. Modelos 3d del aseo ____ *86*
Figura 51. Modelos 3d de la cocina ____ *87*
Figura 52. Modelos 3d del salón ____ *88*
Figura 53. Modelos 3d del pasillo ____ *89*
Figura 54. Modelos 3d de la galería ____ *90*
Figura 55. Modelos 3d de la habitación secreta ____ *91*
Figura 56. Unwrap del teclado del dormitorio 1 ____ *92*
Figura 57. Interfaz de Substance Painter con el modelo de la puerta principal ____ *93*
Figura 58. Página substance share ____ *94*
Figura 59. Puerta principal texturizada ____ *94*
Figura 60. Exportación de texturas ____ *95*
Figura 61. Texturas para la puerta principal ____ *96*
Figura 62. Creación del material de la puerta principal con las texturas anteriores ____ *97*
Figura 63. Mapeo de movimiento y cámara ____ *98*
Figura 64. Blueprints movimiento y cámara ____ *99*
Figura 65. Estructura del blueprint del personaje ____ *100*
Figura 66. Trigger de la puerta del dormitorio 1 ____ *101*
Figura 67. Blueprint del widget interactuar ____ *101*
Figura 68. Widget interactuar ____ *102*
Figura 69. Blueprint de la puerta del dormitorio 1 ____ *102*
Figura 70. Matinee de la animación de una puerta ____ *103*
Figura 71. Mapeado de teclas de la acción interacción ____ *103*
Figura 72. Función principal que controla la inspección de objetos ____ *104*
Figura 73. Función que se encarga del raytracing ____ *104*
Figura 74. Objeto inspeccionable ____ *105*
Figura 75. Mapeo de teclas para la acción inspeccionar ____ *105*
Figura 76. Blueprint de la función enfocar ____ *106*
Figura 77. Mapeado de la acción enfocar ____ *106*
Figura 78. Blueprint de la función de la linterna ____ *107*
Figura 79. Mapeado de la acción linterna ____ *107*
Figura 80. Blueprint principal para dibujar bordes ____ *108*
Figura 81. Comprobación del objeto al que mira el jugador ____ *108*
Figura 82. Función para coger una pieza y dibujarla en el cuadro de la puerta secreta ____ *109*
Figura 83. Algunos sound waves utilizados para el juego ____ *111*
Figura 84. Propiedades de las sound waves ____ *112*
Figura 85. Propiedades de la atenuación del sonido ____ *112*
Figura 86. Blueprint usado para reproducir sonidos ____ *113*
Figura 87. Blueprint para ejecutar un cierto sonido y diferentes propiedades ____ *113*

Figura 88. Pasos en superficie de madera _____ *114*
Figura 89. Pasos en superficie de cemento _____ *115*
Figura 90. Comprobación de si el personaje se mueve o no _____ *115*
Figura 91. Declaración de los materiales físicos _____ *116*
Figura 92. Asignación del material físico al suelo de madera _____ *116*
Figura 93. Blueprints para diferenciar entre superficies y reproducir el sonido deseado _____ *117*
*Figura 94. De derecha a izquierda: video, mediaplayer, textura de video*_____ *118*
*Figura 95. Apariencia del mediaplayer*_____ *118*
Figura 96. Material para pantallas _____ *119*
Figura 97. Vista del material para las pantallas _____ *119*
Figura 98. Imagen del menú principal _____ *122*
Figura 99. Botones menú _____ *122*
Figura 100. Widget del menú principal _____ *123*
*Figura 101. Imagen de los controles del juego*_____ *124*
Figura 102. Imagen de la pantalla de carga _____ *125*
*Figura 103. Imagen InGame del menú de opciones*_____ *126*
*Figura 104. Widget del menú de opciones*_____ *127*
Figura 105. Variables públicas para la configuración _____ *127*
*Figura 106. Lista de comandos para la calidad de las sombras*_____ *127*
Figura 107. Blueprint para aplicar la configuración deseada _____ *128*
Figura 108. Declaración de la cinemática inicial en los ajustes del proyecto _____ *129*
*Figura 109. Captura cinemática inicial*_____ *129*
Figura 110. Blueprint para ejecutar los créditos _____ *130*
Figura 111. Captura de los créditos _____ *130*
*Figura 112. Captura InGame de Psycho1*_____ *135*
Figura 113. Captura InGame de Psycho 2 _____ *135*
Figura 114. Captura InGame de Psycho 3 _____ *136*
Figura 115. Captura InGame de Psycho 4 _____ *136*
Figura 116. Captura InGame de Psycho 5 _____ *137*
Figura 117. Captura InGame de Psycho 6 _____ *137*
Figura 118. Captura InGame de Psycho 7 _____ *138*
Figura 119. Captura InGame de Psycho 8 _____ *138*
Figura 120. Captura InGame de Psycho 9 _____ *139*
Figura 121. Captura InGame de Psycho 10 _____ *139*
Figura 122. Captura InGame de Psycho 11 _____ *140*
Figura 123. Captura InGame de Psycho 12 _____ *140*
Figura 124. Captura InGame de Psycho 13 _____ *141*
Figura 125. Captura InGame de Psycho 14 _____ *141*
Figura 126. Captura InGame de Psycho 15 _____ *142*
Figura 127. Captura InGame de Psycho 16 _____ *142*
Figura 128. Captura InGame de Psycho 17 _____ *143*
Figura 129. Captura InGame de Psycho 18 _____ *143*
Figura 130. Captura InGame de Psycho 19 _____ *144*
Figura 131. Captura InGame de Psycho 20 _____ *144*
Figura 132. Captura InGame de Psycho 21 _____ *145*

Figura 133. Captura InGame de Psycho 22 _____ *145*
Figura 134. Captura InGame de Psycho 23 _____ *146*
*Figura 135. Captura aérea del nivel*_____ *146*

1.Introducción

Es un hecho que, hoy en día, la industria del videojuego se encuentra en uno de sus mejores momentos. Cada día salen cientos de títulos que marcan la vida de una gran cantidad de personas, ya sea reflejando situaciones en las que se ven identificadas o simplemente haciéndoles pasar un rato divertido.

Los videojuegos son, año por año, una de las principales industrias del arte y el entretenimiento, en términos generales son aplicaciones interactivas que nos ofrecen disfrutar de una historia, con nuestra participación en el desarrollo de esta.

Como en todos los sectores, hay distintos campos de trabajo, como por ejemplo la programación, el diseño del videojuego, o el apartado grafico del mismo, entre muchos otros.

Con **Psycho** busco enfrentarme a todos los roles en la creación de un videojuego, pero sobre todo en el ámbito artístico, con la creación integra de todos los *assets*[1] del videojuego.

[1] Cada uno de los elementos que componen el juego como animaciones, modelos, IA, sonidos, etc.

2. Marco teórico o Estado del arte

En este apartado se hablará sobre los videojuegos en la actualidad de una forma más profunda y se comentarán algunas referencias de videojuegos que han despertado una curiosidad en mí.

2.2¿Qué es un videojuego?

Hoy en día es raro encontrar a alguien que no sepa lo que es un videojuego, ya que los niños nacen en un mundo donde la tecnología forma parte de sus vidas desde el primer momento.

Buscando un poco por internet podemos encontrar la siguiente definición:

"Un videojuego es una aplicación interactiva orientada al entretenimiento que, a través de ciertos mandos o controles, permite simular experiencias en la pantalla de un televisor, una computadora u otro dispositivo electrónico."

Los videojuegos pueden ser muy distintos entre sí, tanto en complejidad como en calidad gráfica y en temática. Así como ocurre con el cine y la música, existe una larga y compleja lista de géneros y subgéneros, y la clasificación de un mismo título puede variar según quien lo analice.

Veamos a continuación algunos de los principales géneros:

- Plataformas: se trata de una experiencia que gira en torno a desafíos de tipo físico, que exigen un gran nivel de precisión por parte de los jugadores para avanzar a través de complejas estructuras, generalmente enfrentándose a diversos enemigos. El juego de plataformas por excelencia es Super Mario Bros, desarrollado y publicado por Nintendo en el año 1985; su diseño, su dirección y su producción estuvieron a cargo de *Shigeru Miyamoto*, una eminencia en el mundo de los videojuegos.

- Disparos: un género amplio, al cual pertenecen tanto algunos títulos de guerra como de naves espaciales. Su nombre en inglés es *shooters* y, si bien hace alusión a la acción de disparar, no debe tratarse necesariamente de un arma de fuego, ya que cualquier juego que centre el progreso del personaje principal en la utilización de algún poder que sea despedido hacia los enemigos, ya sea en forma de proyectil o de rayo (entre otras muchas posibilidades), puede entrar en esta categoría.

- Aventuras: son juegos en los cuales la historia es la protagonista. Deben ser construidos en base a una rica narrativa, que atrape poco a poco al jugador en el mundo virtual, y le transmita la necesidad de resolver una serie de misterios. Por lo general, contienen una gran variedad de elementos que resultan fundamentales para avanzar, y estos suelen encontrarse en el escenario mismo, lo cual invita a la exploración constante.

- Rol: suelen confundirse con los juegos de aventuras, pero a diferencia de estos últimos, su foco son los personajes y su evolución a lo largo de la historia. Este género es especialmente popular en Japón, aunque existen muchas comunidades de jugadores de rol en todas partes del mundo. Una de sus características principales es la utilización del término nivel para referirse al grado de experiencia de sus protagonistas y no a los diferentes mundos y escenarios que deben atravesar.

- Deportivos: si bien su nombre parece decirlo todo, existe una línea muy delgada entre este género y el de simulación. Estos dos, a su vez, están emparentados con el género de acción; todo depende del grado de realismo, de la experiencia y del tipo de interacción que se espere del jugador, entre otros factores. Un juego de deportes refleja fielmente las reglas de la disciplina original, pero no a niveles milimétricos, sino que se vale de ciertas licencias, como ser que el tiempo avance más rápidamente que en la realidad.

El rol social de los videojuegos suele estar en discusión. En principio eran considerados como un divertimento para niños y adolescentes, pero la franja etaria se ha ampliado considerablemente en los últimos años. Muy a menudo los videojuegos son vistos como una pérdida de tiempo y una fuente de distracción, especialmente por personas que jamás los han probado; por otro lado, muchos expertos destacan sus valores educativos y pedagógicos. [1]

2.2. Referencias

A continuación, repasaremos algunos de los títulos del género de terror que me han inspirado a la hora de crear ***Psycho***, pero antes entraremos en detalle con el género de terror.

El horror de supervivencia (en inglés *survival horror*) es un género de videojuegos enfocado principalmente a aterrorizar y atemorizar al jugador, con lo que se pretende provocar inquietud, desasosiego o incluso miedo. Estos videojuegos, encuadrados dentro del género de la acción-aventura, hacen uso de los temas, clichés, recursos estéticos y narrativos propios del cine y la novela de terror, potenciándolos a través de la capacidad de inmersión que caracteriza al medio. Así, elementos argumentales como fantasmas, casas encantadas, pueblos malditos, zombis, posesiones demoníacas o alienígenas hostiles son habituales, prefiriéndose el terror de origen sobrenatural sobre otros tipos más realistas.

Desde el punto de vista narrativo, el horror de supervivencia se caracteriza por situar al jugador en algún escenario lúgubre, claustrofóbico, perturbador y, por lo general, escasamente iluminado. Al principio se suele saber poco o nada del lugar en el que se encuentra el personaje y por el que debe transitar con algún pretexto —que puede ser tan simple como la supervivencia— mientras es acechado por criaturas o entes hostiles de aspecto terrorífico. Además, a lo largo de su periplo el jugador encontrará documentos que aclararán poco a poco la historia, dando sentido a la trama.

Otros elementos que definen el género son la tendencia a limitar los encuentros de combate, limitando el acceso a armas o incluso eliminando totalmente la posibilidad de defenderse —forzando al jugador a optar por el sigilo o la huida—, y la presencia periódica de determinados enigmas o puzles de dificultad variable que el jugador deberá resolver para avanzar. [2]

2.2.1. *Emily Wants To Play*

Emily Wants to Play es un videojuego de terror de supervivencia creado por el desarrollador independiente *Shawn Hitchcock*.

El jugador toma el papel de un repartidor de pizza que ha sido atrapado en una casa por una joven llamada *Emily* y sus tres muñecas. Cada nivel del juego está representado por una hora diferente de la noche desde las 11pm hasta las 6am. Durante estos niveles, aparecen varias combinaciones de las muñecas y *Emily*. El jugador debe aprender a interactuar con *Emily* y sus muñecas para sobrevivir a la noche. [3]

Figura 1. Imagen in-game de Emily Wants To Play

Lo que más me ha llamado la atención de este juego han sido las diferentes formas de actuar ante las situaciones a las que te enfrenta el juego. Si no aprendes las mecánicas de cada enemigo, puede ser fatal.
También me ha encantado el apartado visual, ya que se acerca bastante a la estética que buscaba para ***Psycho***.

2.2.2. *Play With Me*

El jugador se despierta en una misteriosa casa vacía y necesita encontrar una manera de escapar evitando al mismo tiempo el contacto directo con el antagonista del juego (un payaso).

Play With Me es un proyecto de un estudiante francés con el objetivo de una aventura de terror de inmersión en la que se deben recoger una serie de velas de cumpleaños y resolver unos acertijos para salir de la casa. [4]

Figura 2. Imagen in-game de Play With Me

Pese a que es un juego bastante corto y los gráficos son bastante simples, consigue crear una tensión constante en el jugador que hace que no estés demasiado tiempo sin hacer nada.

2.2.3. P.T

P.T. (abreviatura de *playable teaser*) es un videojuego de terror psicológico en primera persona desarrollado por *Kojima Productions*, bajo el seudónimo de *7780s Studio*, y publicado por *Konami*. El juego fue dirigido y diseñado por *Hideo Kojima*, en colaboración con el director de cine Guillermo Del Toro.

A diferencia de la perspectiva de tercera persona en los juegos de *Silent Hill*, P.T. utiliza una perspectiva en primera persona, que se centra en un protagonista desconocido, a quien el jugador controla, que despierta en una casa suburbana embrujada y experimenta ocurrencias sobrenaturales. Las áreas disponibles para explorar en la casa consisten en un corredor en forma de L con dos habitaciones adyacentes: un baño y una escalera que conduce a la habitación en la que el jugador inicia un bucle, o una reencarnación continua del corredor.

La única acción que el jugador puede usar es caminar. Para progresar, el jugador debe investigar eventos aterradores y resolver puzles crípticos. Cada vez que un bucle se completa con éxito, los cambios aparecen en el corredor. Además, el jugador se encuentra con un fantasma hostil llamado Lisa. Si capta al protagonista, el jugador tiene una posibilidad aleatoria de desencadenar un sorprendente susto al girar la cámara horizontalmente y es devuelto al principio del bucle. [5]

Figura 3. Imagen in-game de P.T.

Al ser una demo es una experiencia bastante corta, pero fue una demostración del potencial que se podía conseguir con un elemento tan simple como un pasillo.

2.2.4. *Layers of fear*

Layers of Fear es un videojuego de terror psicológico desarrollado y publicado por *Bloober Team* para *Linux, Microsoft Windows, OS X, PlayStation 4 y Xbox One* que se lanzó en todo el mundo el 16 de febrero de 2016.

El jugador controla a un pintor psicológicamente perturbado que está tratando de completar su obra magna, mientras navega a través de una mansión victoriana, con secretos inquietantes sobre el pintor. La jugabilidad, presentada en primera persona, se basa principalmente en la historia y, a menudo gira en torno a la resolución de rompecabezas y la exploración, ya que el juego se intensifica después de cada nivel, mientras que los *jump scares*[2] se producen con frecuencia.

El desafío proviene de acertijos que requieren que el jugador busque en el entorno pistas visuales. La casa parece sencilla al principio, pero cambia alrededor del jugador mientras lo explora. Estos cambios en el entorno proporcionan una continuación para los acertijos y brindan *jump scares* o sustos comunes en los juegos de este género. [6]

Figura 4. Imagen in-game de Layers of Fear.

[2] Técnica, típicamente utilizada en películas de terror y videojuegos, de tener algo que ocurra repentinamente y sin previo aviso para asustar al espectador.

Desarrollado por un equipo independiente está a la altura de juegos punteros del mercado. El realismo de la escena es increíble y la jugabilidad que presenta te introducen en la escena y te incitan a explorar todo lo que puedas de ella.

2.2.5. *Resident Evil* 7

Resident Evil 7 es un videojuego perteneciente al género del survival horror, desarrollado por la empresa japonesa *Capcom*. Es el undécimo título de la serie principal de *Resident Evil* y, a diferencia de la mayoría de los otros juegos de la franquicia, es en primera persona, siendo el primero de la saga principal en primera persona.

A diferencia de otros videojuegos más cercanos al género del horror puro, el jugador controla a *Ethan Winters*, quien a diferencia de los protagonistas de los otros *Resident Evil*, es solo un civil con pocas habilidades especiales de lucha, y sin ningún entrenamiento de combate policial o militar, aun así, es capaz de usar un considerable repertorio de armas (tanto armas de tipo cuerpo a cuerpo como de disparos), y tiene la opción de oponerse y luchar contra los distintos enemigos.

Además, el jugador al igual que en las últimas entregas, puede girar rápidamente 180 grados para localizar y evitar a los enemigos, así como por primera vez en la saga principal se tiene la posibilidad de bloquear los ataques de los enemigos para reducir el daño. Varios tramos del juego consisten en ser acechados por los miembros de la familia *Baker*, quienes no pueden ser aniquilados en los primeros combates, solo pueden ser temporalmente incapacitados. Sin embargo, estos encuentros son evitables mediante el sigilo o huyendo.

Otros elementos populares de la serie *Resident Evil* como los rompecabezas, la gestión de recursos en los baúles de almacenamiento, la mezcla y combinación de distintos recursos, las pólvoras, y las hierbas curativas también están disponibles en este título.

A diferencia de anteriores *Resident Evil*, el modo de juego enfatiza más el horror y la exploración sobre la acción. Muchos de los rompecabezas del juego requieren que los elementos sean examinados bajo ciertas condiciones para revelar secretos. Las grabadoras se pueden usar para guardar manualmente el progreso del juego, lo que, dependiendo del nivel de dificultad, puede requerir el uso de una pequeña cinta de casete. Existen otras cintas, las de vídeo VHS que se encuentran dispersas para que *Ethan* las encuentre, estas cintas sirven para activar misiones o *flashbacks*, que colocan al jugador en la perspectiva de un personaje diferente, a menudo revelando información de la trama, o pistas necesarias para resolver un rompecabezas. [7]

Figura 5. Imagen in-game de Resident Evil 7

Para culminar con las referencias tenemos un juego que estuvo en boca de muchos. No solo se separó del sistema con el que contaban los anteriores *Resident Evil*, sino que consiguió que muchas personas que no conocían este género se interesasen por él.

Gráficamente no deja nada que desear, y su diseño artístico es el que más me ha inspirado en la creación de Psycho.

3. Objetivos

El objetivo de este proyecto es la creación de un videojuego de terror para ordenador, enfocándose en la parte gráfica, ya que todos los modelos que se usarán para el mismo se realizarán de cero, sin usar recursos externos.

Se usará el motor gráfico *Unreal Engine* 4, en su versión 4.18 y la programación de las mecánicas se hará mediante el sistema visual *Blueprints*.

También se realizará un documento GDD del videojuego, se realizarán análisis de juegos del mismo género y se detallará como se han realizado paso a paso algunos de los modelos creados para el videojuego.

3.1. Desglose de objetivos

A continuación, se muestran los objetivos del proyecto de forma más detallada:

1. Realizar el Documento de Diseño de un videojuego (GDD).
2. Creación completa de *assets* para un videojuego.
3. Creación de un entorno realista utilizando las herramientas que ofrece *Unreal Engine 4*.
4. Completar la lógica del videojuego usando el sistema de *Blueprints*.
5. Sonorización realista de un videojuego.
6. Creación de menús, interfaz y opciones gráficas de un videojuego.

4. Metodología

4.1. Kanban

Kanban se basan en el desarrollo incremental, dividiendo el trabajo en partes. Una de las principales aportaciones es que utiliza técnicas visuales para ver la situación de cada una de las tareas, normalmente representadas en pizarras llenas de *post-it*.

El trabajo se divide en partes, normalmente cada una de esas partes se escribe en un *post-it* y se pega en una pizarra. Los *post-it* suelen tener información variada, si bien, aparte de la descripción, debieran tener la estimación de la duración de la tarea.

La pizarra tiene tantas columnas como estados por los que puede pasar la tarea (ejemplo, en espera de ser desarrollada, en análisis, en diseño, etc.).

El objetivo de esta visualización es que quede claro el trabajo a realizar, en qué está trabajando cada persona, que todo el mundo tenga algo que hacer y el tener clara la prioridad de las tareas.

Las fases del ciclo de producción o flujo de trabajo se deben decidir según el caso, no hay nada acotado. [9]

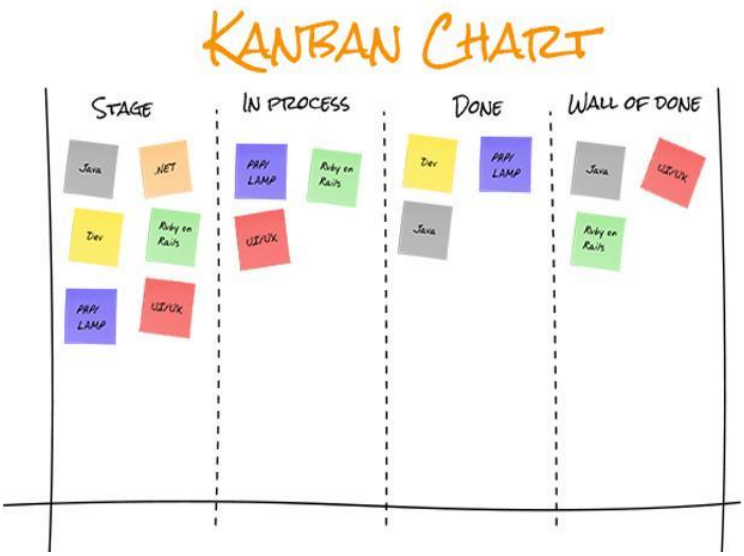

Figura 6. Imagen explicativa de la metodología Kanban

4.2. Gestión del proyecto

Para la gestión del proyecto se ha utilizado tanto la herramienta *Trello*, la cual permite ordenar tarjetas con diferentes tareas de forma virtual, así como una pizarra en la que gestionaba las tareas de forma parecida a *Trello*.

Posteriormente se usó únicamente la pizarra ya que permitía ver todas las tareas de una manera más rápida, así como redistribuir las tareas en caso de que fuera necesario.

4.3. Control de versiones

Debido al gran tamaño del proyecto, no se ha podido utilizar herramientas de trabajo en la nube como *Git* o similares, por lo que se ha optado por realizar copias de seguridad regularmente en diferentes dispositivos.

5. Cuerpo del trabajo

En este apartado se hablará en detalle del proceso de la creación del videojuego ***Psycho***, que se dividirá en las siguientes partes:

- Un primer apartado donde se hablará de las herramientas que se ha considerado su uso para el proyecto, así como las seleccionadas.
- Un segundo apartado donde se encuentra el GDD (Documento de Diseño del Videojuego), en el que se hablará de las características del videojuego, así como todo el proceso de creación de este.

5.1. Elección de herramientas

5.1.1. Motores Gráficos

A continuación, hablaremos sobre diferentes motores gráficos que se han tenido en cuenta para la realización de este proyecto. Cabe destacar que hay cientos de motores, tanto de uso privado, como de uso comercial, por lo que nos centraremos en aquellos de uso comercial más usados en la industria.

5.1.1.1. *Unity*

Unity es un motor de videojuego multiplataforma creado por *Unity Technologies* y está disponible como plataforma de desarrollo para *Microsoft Windows*, *OS X* y *Linux*.

Unity puede usarse junto a una gran variedad de software como *Blender*, *3ds Max*, *Maya*, *Softimage*, *Modo*, *ZBrush*, *Adobe Photoshop*, etc.

Los cambios realizados a los objetos creados con estos productos se actualizan automáticamente en todas las instancias de ese objeto durante todo el proyecto sin necesidad de volver a importar manualmente, por lo que nos permite ahorrar tiempo y avanzar rápidamente.

Unity destaca sobre todo en el sector móvil, aunque poco a poco va adentrándose en los diferentes dispositivos de la actualidad, como lo son los ordenadores personales o las consolas. [10]

5.1.1.2. *Unreal Engine*

Unreal Engine es un motor de juego de PC y consolas creado por la compañía *Epic Games*, demostrado inicialmente en el shooter en primera persona *Unreal* en 1998, y siendo la base de grandes juegos como *Unreal Tournament*, *BioShock* o *Paragon*, entre otros.

También se ha utilizado en otros géneros como el rol y juegos de perspectiva en tercera persona.

Unreal Engine ha tenido cuatro versiones hasta la fecha, tres de ellas de pago y la cuarta, desde el año 2015 gratuita para todo el mundo.

La versión actual, *Unreal Engine* 4, está diseñada para las plataformas *Microsoft Windows, macOS, Linux, SteamOS, HTML5, iOS, Android, PlayStation4, Nintendo Switch, Xbox One SteamVR/HTC Vive, Oculus Rift, PlayStation VR, Google Daydream, OSVR* y *Samsung Gear VR*. [11]

5.1.1.1. Elección del motor

La elección del motor es algo crucial, ya que nuestro resultado final puede cambiar drásticamente dependiendo del motor que usemos.

Debido a la naturaleza del proyecto y tras estudiar estas dos opciones junto a algunos otros motores, he decidido escoger *Unreal Engine*.

Esto se debe a que es un motor muy utilizado por las empresas de videojuegos más famosas de la industria y tiene gran cantidad de documentación en línea, siendo su propio foro uno de los más activos de todos. También cuenta con una gran cantidad de contenido en *YouTube* realizado por múltiples personas que han sido de gran ayuda durante el desarrollo de este proyecto.

Unreal Engine cuenta con herramientas que son 100% gratuitas y solo nos pide un porcentaje de nuestros beneficios si superamos ciertos umbrales anuales.

Otra de las razones para elegir *Unreal Engine* es su sistema de programación visual mediante *blueprints*, el cual veremos más adelante, ya que nos permite programar de una manera intuitiva y enfocando nuestro esfuerzo en la tarea de modelado de este proyecto.

Por último, también nos permite un acabado final que pocos motores ofrecen hoy en día, todo ello de una manera bastante intuitiva y visual.

5.1.2. Software de Modelado

A continuación, hablaremos sobre diferentes programas de modelado que se han tenido en cuenta para la realización de este proyecto.

5.1.2.1. *Blender*

Blender es un programa dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. También de composición digital utilizando la técnica procesal de nodos, edición de vídeo, escultura (incluye topología dinámica) y pintura digital.

El programa fue inicialmente distribuido de forma gratuita, pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de *Windows*, *Mac OS X*, *GNU/Linux* (Incluyendo Android), *Solaris*, *FreeBSD* e *IRIX*.

Blender, a pesar de ser gratuito, no se queda corto en nada, ya que está a la altura de *3ds Max* y es usado por muchas pequeñas empresas en la industria del cine y los videojuegos. No obstante, no tengo tanta experiencia con él, así que no se usará en este proyecto. [12]

5.1.2.2. *Maya*

Autodesk Maya es un programa informático dedicado al desarrollo de gráficos 3D por ordenador, efectos especiales y animación.

Surgió a partir de la evolución de *Power Animator* y de la fusión de *Alias* y *Wavefront*, dos empresas canadienses dedicadas a los gráficos generados por ordenador. Más tarde *Silicon Graphics* (ahora SGI), el gigante informático, absorbió a *Alias-Wavefront*, que finalmente fue absorbida por *Autodesk* dueña de *3d Studio Max*, por la cantidad de 182 millones de dólares.

Maya se caracteriza por su potencia y las posibilidades de expansión y personalización de su interfaz y herramientas. MEL (*Maya Embedded Language*) es el código que forma el núcleo de *Maya* y gracias al cual se pueden crear *scripts* y personalizarlo.

El programa posee diversas herramientas para modelado, animación, renderización, simulación de ropa y cabello, dinámicas (simulación de fluidos), etc. [13]

Maya es una herramienta muy similar a *3ds Max*, más ligera y especializada en técnicas de animación.

5.1.2.3. 3ds Max

Autodesk 3ds Max es un programa de creación de gráficos y animación 3D desarrollado por *Autodesk*, en concreto la división *Autodesk Media & Entertainment* (anteriormente *Discreet*). Creado inicialmente por el Grupo *Yost* para *Autodesk*, salió a la venta por primera vez en 1990 para DOS.

3ds Max, con su arquitectura basada en *plugins*, es uno de los programas de animación 3D más utilizado, especialmente para la creación de videojuegos, anuncios de televisión, en arquitectura o en películas. [14]

Se ha optado por este *software* debido a que se ha usado en la carrera y estoy más familiarizado a su interfaz y técnicas de modelado, por lo que el desarrollo del proyecto se llevará a cabo de una forma óptima.

5.1.3. Texturizado

5.1.3.1. Mudbox

Mudbox es un *software* de modelado 3d, texturizado y pintura digital, actualmente desarrollado por *Autodesk*. *Mudbox* fue creado por *Skymatter*, fundado por artistas de *Weta Digital*, y fue usado por primera vez en la película *King Kong* (película de 2005). [15]

Mudbox tiene un gran uso en la industria, aunque otros programas de texturizado tienen más peso hoy en día en la industria, como *Substance Painter*.

5.1.3.2. Substance Painter

Substance Painter es un programa de texturizado, similar a *Mudbox*, que se encuentra en crecimiento y cada vez se usa más en la industria del cine y del videojuego. Cuenta con infinidad de materiales y texturas, las cuales pueden crear la propia comunidad de usuarios y compartirla de forma gratuita.

He decidido usar este programa ya que tengo algo de experiencia con él y permite obtener materiales de muy buena calidad y una muy buena compatibilidad con *Unreal Engine*, como veremos más adelante.

5.1.3.3. *Photoshop*

Adobe Photoshop es un editor de gráficos rasterizados desarrollado por *Adobe Systems Incorporated*. Es usado principalmente para el retoque de fotografías y gráficos y su nombre en español significa literalmente "taller de fotos".

Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general. [16]

He usado esta herramienta junto con *Substance Painter* para llegar a unos mayores niveles de detalle, consiguiendo resultados asombrosos.

5.1.4. Sonido

5.1.4.1. *FreeSound*

Freesound tiene como objetivo crear una gran base de datos colaborativa de fragmentos de audio, muestras, grabaciones, *bleeps*, etc., publicados bajo licencias de *Creative Commons* que permiten su reutilización.

En algunos casos el autor del sonido puede pedir que se le cite en el uso de este.

Freesound ofrece una gran cantidad de sonidos y de muy buena calidad proporcionado por millones de usuarios. [17]

5.1.4.2. *Audacity*

Audacity es *software* libre, desarrollado por un grupo de voluntarios y distribuido bajo la Licencia Pública General de GNU (GPL).

Audacity es un editor y grabador de audio multipista, fácil de usar y gratuito para *Windows*, *Mac OS X*, *GNU / Linux* y otros sistemas operativos. La interfaz está traducida a muchos idiomas. [18]

La mayoría de los sonidos obtenidos en *Freesound*, se han editado y ajustado a las necesidades del proyecto en *Audacity*, así como la grabación propia de algunos sonidos.

5.2. Documento de Diseño del Videojuego (GDD)

El Documento de Diseño del Videojuego, o comúnmente conocido con las siglas en inglés GDD, es un documento que se elabora durante la fase de preproducción de un videojuego y que se va actualizando conforme avanza su desarrollo. Contiene información detallada sobre los componentes y características (objetos, reglas, entornos, estructura narrativa, etc.) del juego, sirviendo como guía a los diferentes grupos de trabajo durante el proceso de desarrollo. [19]

En este apartado se describe el GDD del videojuego realizado.

5.2.1. Conceptos del juego

5.2.1.1. Argumento

Apareces en una casa abandonada, pero con signos de que hace no mucho ahí vivía gente. No hay demasiada luz así que sacas el móvil e intentas saber dónde estás y cómo salir de ahí. Notas algo que te acecha, pero no sabes muy bien que es. Investigas y empiezas a reconocer a la gente que habitaba ahí. Ahora solo tienes que llegar a la salida.

Los dueños de la casa tuvieron una niña. La niña tiene telequinesia, lo cual al principio era algo insignificante, pero poco a poco era algo incontrolable. Los padres, que eran muy creyentes empezaron a tener miedo de ella y buscaron ayuda.

Entre muchas visitas a diferentes instituciones, fueron a un psicólogo, el cual les dijo que debían buscar una solución, ya que cuando crezca podría ser una amenaza.

Los padres, cegados por su devoción, llevaron las palabras del psicólogo más allá, y ante el miedo y el no saber qué hacer, la encerraron y decidieron ocultarla para evitar que pudiera hacer daño a alguien. Podían pasar días sin que recordaran que estaba encerrada, hasta que un día, la encontraron sin vida en el suelo.

Como consecuencia, el espíritu de la niña quedo atrapado en esa casa. No se sabe el paradero de los padres. Tu eres el psicólogo, que sin tener realmente culpa de nada, serás el objetivo de su ira.

El título del juego, ***Psycho***, nace del mismo jugador, el cual es un psicólogo junto con la temática paranormal de la que trata el juego.

5.2.1.2. Objetivo del juego

El objetivo del juego es descubrir qué ha pasado en esa casa, así como intentar salir de ella.

El juego tiene dos finales, en uno puedes intentar salir de la casa cogiendo las llaves de la puerta principal (desapareciendo la otra opción), y en el otro quedar atrapado para siempre en la habitación secreta.

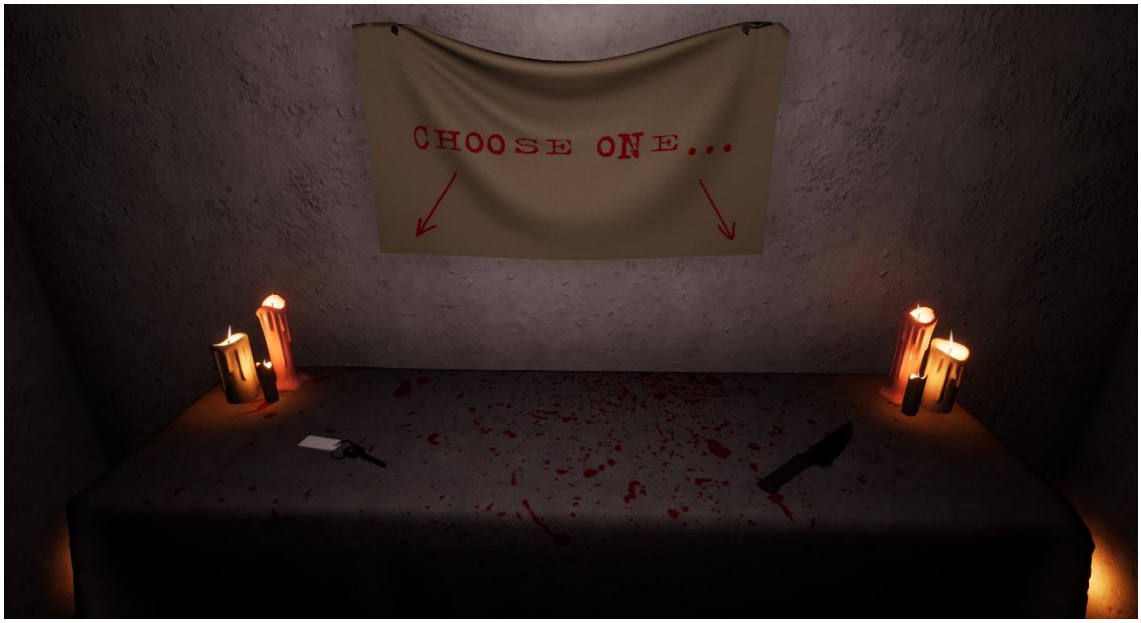

Figura 7. Elección de los dos finales del juego

5.2.1.3. Conjunto de características

La principal característica del juego es la ambientación, que te lleva a una sensación de inmersión y tensión constante, junto al sonido que también ayudará a incrementar dicha característica.

La idea es que el jugador recorra la casa para saber porque se encuentra en esa casa e ir descubriendo la historia de esta, así como intentar salir y escapar de aquello que lo persigue, utilizando mecánicas fáciles de aprender.

5.2.1.4. Género

Psycho pertenece al género ***survival horror***, anteriormente comentado**.** Es una aventura en primera persona inspirada en muchos juegos *indies* que han tenido un gran éxito.

5.2.1.5. Plataforma

El videojuego será desarrollado para ordenador como principal plataforma, así como posteriormente, compatibilizarlo con el uso de gafas de realidad virtual como *Oculus Rift* o similares.

5.2.1.6. Tecnología

El ordenador donde se desarrollará el juego tendrá las siguientes características:

Ordenador Sobremesa:

- Procesador *Intel® Core™* i7-4790K 4.0Ghz
- Memoria RAM 16GB DDR3
- Disco duro 2TB + 480GB SSD
- Tarjeta gráfica MSI *GeForce* GTX 980 4GB GDDR5

Ordenador Portátil:

- Procesador *Intel® Core™* i7-5700HQ 3.50 GHz
- Memoria RAM 16GB DDR3
- Disco duro 1TB + 256GB SSD
- Tarjeta gráfica *GeForce* GTX970M, 3GB GDDR5

5.2.1.7. Público

Por su género, Psycho no es recomendado para menores de 16 años. Enfocado a todo tipo de jugadores, pero en especial a aquellos que les apasione el género de terror o suspense.

5.2.2. Jugabilidad

El personaje se mueve por la casa, interactuando con los diferentes objetos que hay a su paso. Los controles están diseñados de manera intuitiva para la mayoría de los jugadores.

Teclado y ratón:

- W: moverse hacia delante.
- A: moverse hacia la izquierda.
- D: moverse hacia la derecha.
- S: moverse hacia atrás.
- F: encender/apagar la linterna.
- E: inspeccionar objetos.
- Botón derecho del ratón: Acercar/Alejar cámara.
- Botón izquierdo del ratón: Interacción.
- Ratón: cámara.

Mando Xbox:

- Joystick izquierdo: moverse.
- Botón B: encender/apagar la linterna.
- Botón X: inspeccionar objetos.
- Botón RB: Acercar/Alejar cámara.
- Botón A: Interacción.
- Joystick derecho: cámara.

5.2.3. Mecánicas

Las mecánicas de juego son una serie de reglas que intentan generar juegos que se puedan disfrutar, que generen una cierta adicción y compromiso por parte de los usuarios, al aportarles retos y un camino por el que discurrir.

En el caso de ***Pyscho,*** el jugador puede abrir puertas, encender o apagar interruptores de la luz, inspeccionar objetos y utilizar la linterna del móvil, todo esto para conseguir una plena experiencia en el *gameplay*.

Pese a que ***Psycho*** cuenta con pocas mecánicas, son tan importantes que sin ellas no sería lo que es.

Veamos en más detalle estas mecánicas.

5.2.3.1. Movimiento y cámara

El movimiento del personaje es un movimiento simple, en tres dimensiones y cuenta con una velocidad de movimiento muy parecida al de una persona normal, muy común en este tipo de género.

El personaje puede moverse libremente por la casa, pero sin correr, para evitar que pierda detalles visuales.

La cámara es en primera persona, limitando la rapidez a la que se puede mover, para mimetizar los movimientos de una persona normal, al igual que pasa con el movimiento. Cuenta con un pequeño efecto de muelle, con lo que se consigue un movimiento suave, aunque el jugador mueva la cámara bruscamente.

Con estos detalles se consigue dar más naturalidad al personaje que manejamos.

Figura 8. Captura InGame del dormitorio1

5.2.3.2. Interacción

En ***Psycho*** he querido tener la menor interacción directa posible con el jugador. Esto es difícil ya que como no hay que decirle nada directamente al jugador, todos los mensajes que queramos decirle tendrán que ser mediante la narrativa visual. La única forma mínima de decirle algo es mediante una pequeña señal visual.

Por todo esto, la interacción del jugador con el entorno es muy simple. Cuando abajo a la izquierda aparece una marca de aviso, significa que el jugador puede interactuar con algo cercano a él, como por ejemplo abrir una puerta, encender las luces o coger objetos especiales.

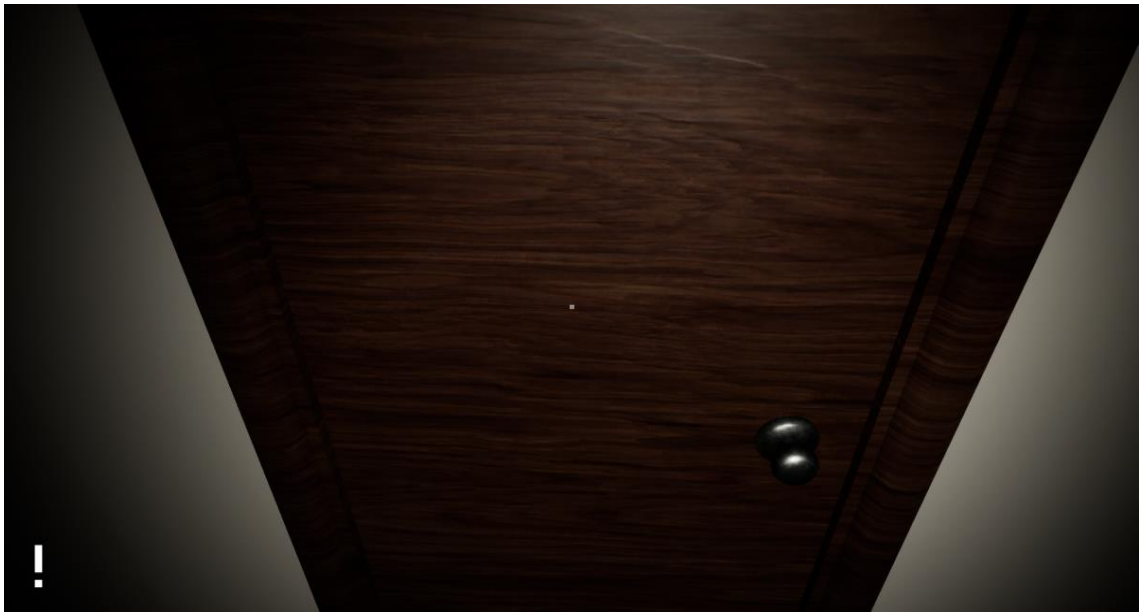

Figura 9. Captura InGame que muestra la interacción con una puerta

5.2.3.3. Inspección

Cualquier elemento que no esté fijado a la pared o al suelo se puede inspeccionar, para ello nos acercamos al objeto en cuestión, apuntamos hacia él con la ayuda de la retícula o punto blanco que tenemos en el centro de la pantalla y pulsamos la tecla de inspección, que en el caso del teclado es la tecla “E”.

Una vez pulsada la tecla, el objeto realiza una animación en la que se nos acerca y con la ayuda del ratón podemos rotar el objeto para verlo en detalle o descubrir pistas ocultas en él si las hubiera.

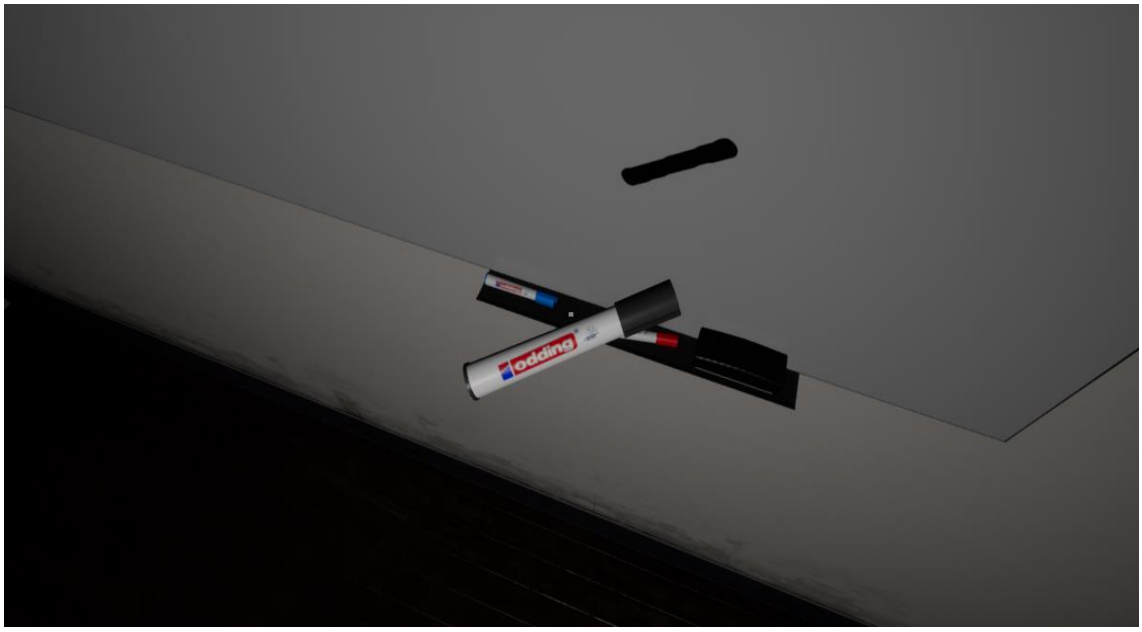

Figura 10. Captura InGame que muestra la inspección de un rotulador

5.2.3.4. Enfoque

Cuando realizamos la acción de enfocar, que en el caso del ratón es el botón derecho, reducimos el campo de visión, enfocando a la zona a la que estemos mirando para obtener un mejor detalle.

También podemos combinar esta mecánica con la inspección de objetos para ver algún detalle de una manera más precisa.

Figura 11. Captura InGame de la cocina sin enfoque

Figura 12. Captura InGame de la cocina con enfoque

5.2.3.5. Linterna

En algunas ocasiones se puede ir la luz, por lo que siempre disponemos de la linterna del móvil para no quedarnos completamente a oscuras. Esto nos proporciona una iluminación muy tenue, simplemente para ver hacia dónde vamos o accionar un interruptor de la luz.

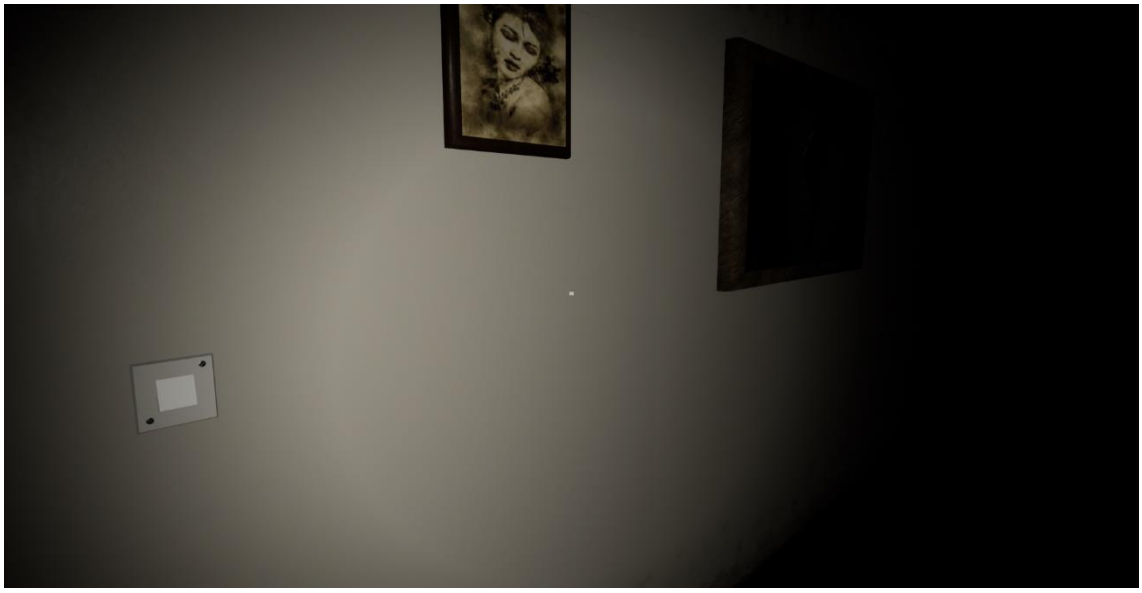

Figura 13. Captura InGame del uso de la linterna en el pasillo

Figura 14. Captura InGame del uso de la linterna en la cocina

5.2.3.6. Piezas Secretas

Es la mecánica más importante en el desarrollo del juego, ya que reuniendo las 4 piezas que hay abriremos la habitación secreta. Al ir cogiendo estas piezas se van añadiendo al cuadro original que está en la puerta de la habitación secreta.

Estas piezas se iluminan de una forma especial para llamar la atención del jugador.

Figura 15. Pieza secreta resaltada al mirarla

Figura 16. Piezas secretas colocadas según se han cogido

5.2.4. Personajes

En el juego solo hay un personaje, el que nosotros mismos manejamos. Por el momento no se incluirá una IA, simplemente habrá eventos y *triggers* con animaciones, ya que se incidirá más en el detalle de la casa y la experiencia en ella.

Aunque no contemos con ninguna IA, los diferentes efectos que transcurren en la casa junto al sonido hacen que el jugador se encuentre en tensión ante la duda de si le puede ocurrir algo o no.

5.2.5. Nivel del juego

Todo el juego, así como todos los eventos que aparecen, transcurren en un solo nivel. Este nivel es una casa normal con un ligero toque americano, expresado en los muebles y el estilo visual de las zonas, donde se pueden apreciar diferentes estilos decorativos.

Cuenta con 8 estancias donde se pueden realizar diferentes acciones. Estos lugares son: el dormitorio 1, el baño, el dormitorio 2, la cocina, el salón, el pasillo, la galería y la habitación secreta.

Muchos de los elementos no están en un cierto lugar porque sí, sino que intentan ofrecer información visual de la historia o del tipo de gente que vive en la casa.

La creación del nivel se verá posteriormente.

Figura 17. Boceto original de la estructura de la casa

5.2.6. Progresión del Juego

Todo el juego transcurre en la misma casa, siendo la progresión del juego la siguiente:

1. El jugador se despierta en el dormitorio 1.
2. Enciende la linterna del móvil y busca el interruptor de la luz (el cual destaca en la oscuridad).
3. Sale del dormitorio y explora la casa.
4. Las luces están apagadas, se pueden encender, pero pueden apagarse en cualquier momento.
5. Busca las piezas para abrir la habitación “secreta”, repartidas por la casa.
6. Descubre la historia de la casa y de lo que habita en ella.
7. Una vez halladas todas las piezas, entra a la habitación secreta.
8. En la habitación secreta:
    a. Coge el cuchillo, va a la salida y fin del juego.
    b. Coge la llave de la casa, se cierra la puerta de la habitación secreta y se queda en la casa para siempre y fin del juego.

5.3. Desarrollo

En este apartado veremos unos conceptos básicos sobre *Unreal Engine* y profundizaremos en el desarrollo del nivel, enfocándonos en los modelos 3d que componen el juego.

También hablaremos sobre el desarrollo de las mecánicas de las que se compone el juego, así como de la parte multimedia y los menús realizados para este proyecto.

5.3.1. *Unreal Engine*

Vamos a ver la interfaz del motor que usaremos para este proyecto. Cuando ejecutamos *Unreal Engine* nos aparece la escena de inicio por defecto, con las siguientes partes:

- En el número 1, tenemos las pestañas, las cuales funcionan de manera similar a un navegador de internet, pudiendo tener varias abiertas en un mismo proyecto e incluso en diferentes pantallas. Al abrir el proyecto siempre nos aparece la interfaz principal.

- En el número 2, tenemos las opciones rápidas del motor, como ejecutar el proyecto, diseñar la iluminación, guardar el nivel, etc. Esta pestaña se usará mucho durante el desarrollo del proyecto.

- En el número 3, tenemos los elementos basicos que utilizaremos, como son geometrías básicas, luces, personajes, *triggers*, etc. También nos ofrece otras opciones como la herramienta de edición de terreno y edición de geometría entre otras. Es muy parecido a un buscador que contiene todo lo que podemos arrastrar a la escena.

- En el número 4, tenemos el explorador de archivos donde encontraremos todo lo que tengamos de base en nuestro proyecto, así como todo lo que importemos. Si lo organizamos todo bien, nos será de gran ayuda.

- En el número 5, tenemos la escena, la cual es donde se construye visualmente nuestro juego. Podemos movernos en ella de forma muy parecida a un programa de edición 3D. También podemos arrastras elementos e incluirlos de forma muy rápida en la escena.

- En el número 6, tenemos un menú que nos muestra todos los componentes de nuestra escena. Podemos agruparlos en carpetas para tenerlo todo más organizado, y nos permite encontrar objetos de forma muy rápida.

- Y, por último, en el número 7, tenemos el panel de detalles. Este panel nos muestra las propiedades del objeto que tengamos seleccionado y nos permite editarlas de una forma muy rápida. [20]

Figura 18. Interfaz de Unreal Engine

También tenemos otras pestañas que son las más usadas como *blueprints*, los cuales veremos más adelante, *widgets*, propiedades del proyecto, etc. Alguna de ellas las iremos viendo en distintas áreas de este documento.

A continuación, veremos algunos de los elementos más usados en este proyecto, así como una explicacion detallada de los mismos.

5.3.1.1. *Blueprints*

El sistema de *blueprints* en *Unreal Engine* es un sistema de *scripting* basado en el concepto de utilizar una interfaz basada en nodos para crear elementos de juego. Al igual que con muchos lenguajes de *scripting* comunes, se usa para definir clases u objetos orientados a objetos en el motor.

A menudo encontraremos que los objetos definidos con *blueprint* se denominan coloquialmente simplemente como *Blueprints*.

Este sistema es extremadamente flexible y poderoso, ya que brinda a los diseñadores la capacidad de utilizar visualmente toda la gama de conceptos y herramientas que generalmente solo están disponibles para los programadores. [21]

5.3.1.1.1. ¿Cómo funcionan?

En su forma básica, los *blueprints* son adiciones visualmente añadidas a un juego. Al conectar nodos, eventos, funciones y variables con cables, es posible crear elementos de juego complejos.

Funcionan mediante el uso de nodos para diversos fines: construcción de objetos, funciones individuales y eventos generales de juego, que son específicos de cada instancia del *blueprint* para implementar el comportamiento y otras funcionalidades.

5.3.1.1.2. Tipos de *blueprints*

Vamos a ver los principales tipos de *blueprints* que tiene *Unreal Engine* y que se han usado en este proyecto. [22]

5.3.1.1.2.1. *Blueprint* de nivel

Cada nivel tiene su propio *blueprint* de nivel, y esto puede hacer referencia y manipular actores dentro del nivel, controlar cinemáticas usando *matinee actors*, y administrar diferentes cosas como transmisión de niveles, puntos de control y otros sistemas relacionados con niveles.

El *blueprint* de nivel también puede interactuar con los *blueprint* de clase, los que veremos a continuación, ubicados en el nivel, como leer o configurar cualquier variable o desencadenar eventos personalizados que pueda contener. [23]

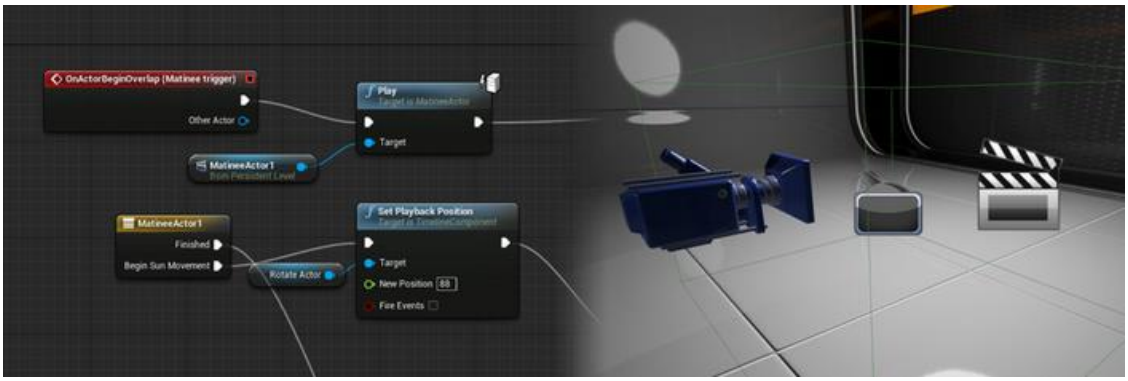

Figura 19. Ejemplo de un blueprint de nivel

5.3.1.1.2.2. *Blueprint* de clase

Las clases de *blueprint* son ideales para hacer elementos interactivos, como puertas, interruptores, artículos coleccionables y escenarios destructibles. En la imagen de abajo, el botón y el conjunto de puertas son cada uno de los *blueprints* que contienen la secuencia de comandos necesaria para responder a los eventos de los jugadores, animarlos, reproducir los efectos de sonido y cambiar sus materiales. [24]

Figura 20. Ejemplo de un blueprint de clase

En este caso, presionar el botón activa un evento dentro de *blueprint* de la puerta, lo que hace que se abra, pero las puertas podrían activarse fácilmente con otro tipo de *blueprint* o una secuencia de *blueprints* del nivel.

Debido a la naturaleza autónoma de los *blueprints*, se pueden construir de tal manera que se puedan colocar en un nivel y simplemente funcionarán, con una configuración mínima requerida.

5.3.1.1.2. ¿Qué más pueden hacer los *blueprints*?

A parte de lo anteriormente comentado, podemos utilizar *blueprints* para muchas otras cosas, como por ejemplo la creación de un jugador que podemos utilizar en diferentes niveles hasta la creación de un HUD, o comúnmente conocido como la interfaz del juego.

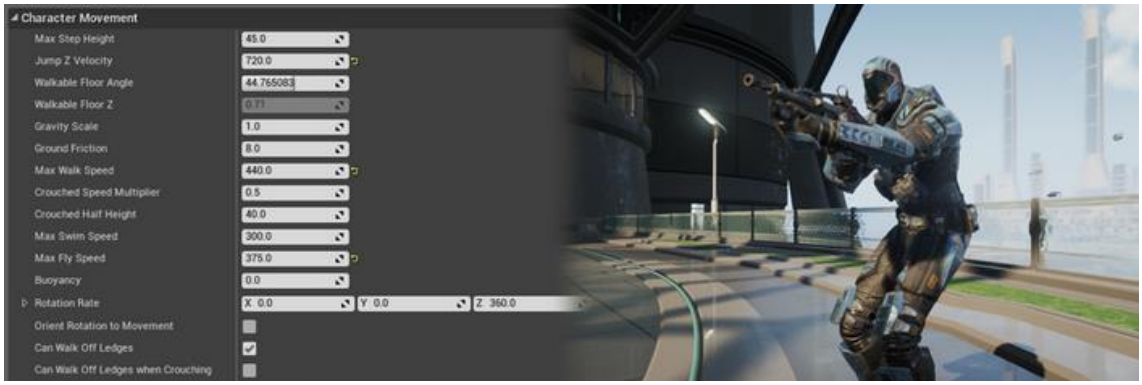

Figura 21. Ejemplo de las opciones del blueprint carácter

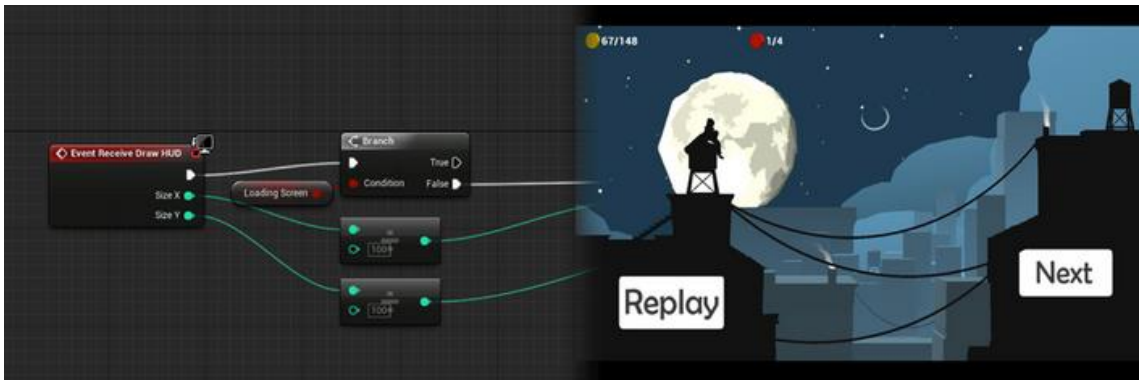

Figura 22. Ejemplo de blueprint de interfaz o HUD

5.3.1.2. Luces

Las luces son la principal herramienta en los videojuegos para conseguir un buen nivel de realismo. Sabiendo qué tipo de luz utilizar en un determinado momento, se pueden conseguir resultados espectaculares.

Antes de comenzar a ver las luces primero debemos comprender el concepto de *lightmap*. Un *lightmap* es una estructura de datos usada en el *lightmapping*, una forma de precalcular el brillo de las superficies y guardarlo en mapas de texturas para su posterior uso. [25]

5.3.1.2.1. Tipos de luces

Unreal engine cuenta con cuatro luces principales: Direccional, Puntual, Foco y *Sky*.

- Direccional: es una luz que parece venir desde el infinito, se suele usar mucho para emular los rayos de sol.

Figura 23. Luz direccional

- Puntual: es la luz más usada, ya que imita a una bombilla, emitiendo la luz desde un punto hacia todas las direcciones.

Figura 24. Luz puntual

- Foco: como su nombre indica se comporta igual que un foco, emitiendo la luz desde un punto de forma cónica.

Figura 25. Luz de foco

- *Sky*: es una luz general que ilumina de forma global, normalmente utilizada para generar una luz de ambiente. [26]

Figura 26. Luz ambiental

5.3.1.2.1. Propiedades de las luces

Cada una de las luces tiene diferentes propiedades como la distancia o la intensidad, pero todas comparten una propiedad muy importante, la movilidad.

La movilidad es una propiedad que tiene un gran impacto en el rendimiento del juego. En *Unreal Engine* tenemos principalmente tres tipos de movilidad para las luces: movilidad estática, movilidad estacionaria y movilidad dinámica:

- Estática: la luz no se puede cambiar en el juego. Este es el método más rápido para renderizar y permite construir la iluminación, lo que significa que podemos calcular las sombras y añadirlas a la textura para evitar tener que calcularlas en tiempo de juego.

- Estacionaria: la luz solo tendrá su sombra y su luz rebotada de la geometría estática construida por *lightmass*, todas las demás serán dinámicas. Esta configuración también permite que la luz cambie de color e intensidad en el juego, pero no se mueve y permite una iluminación parcial construida.

- Movible: la luz es totalmente dinámica y permite un sombreado dinámico. Este es el más lento en términos de rendimiento, pero permite la mayor flexibilidad durante el juego. [27]

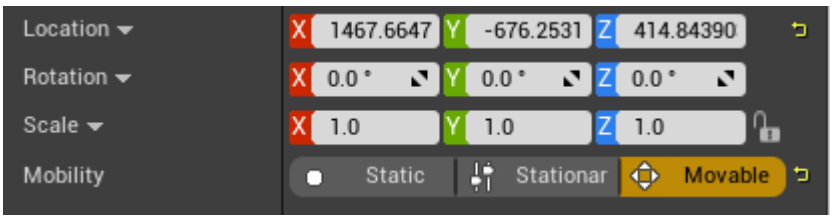

Figura 27. Tipos de movilidad de una luz

En nuestro caso debido al uso de la linterna, la mayoría de las luces usadas en este proyecto son luces puntuales con movilidad de tipo movible. Esto es así debido a que, si usáramos otro tipo de movilidad, al iluminar esa superficie con la linterna se podrían ver sombras indeseadas y también por el hecho de que la mayoría de las luces se pueden encender y apagar durante la ejecución del juego.

5.3.1.3. Materiales

Los materiales son uno de los aspectos más críticos para lograr un buen aspecto para los objetos y niveles. Son aquello que le aporta mayor realismo al modelo, dotándolo de mejoras visuales complementarias al propio nivel de detalle.

Lo primero y más importante que se debe saber sobre los materiales es que no se crean a través de un código, sino a través de una red de nodos visuales de *scripting* dentro del editor de materiales. Cada nodo contiene un fragmento de código HLSL, designado para realizar una tarea específica. Esto significa que a medida que construyes un Material, estás creando código HLSL a través de *scripts* visuales. [28]

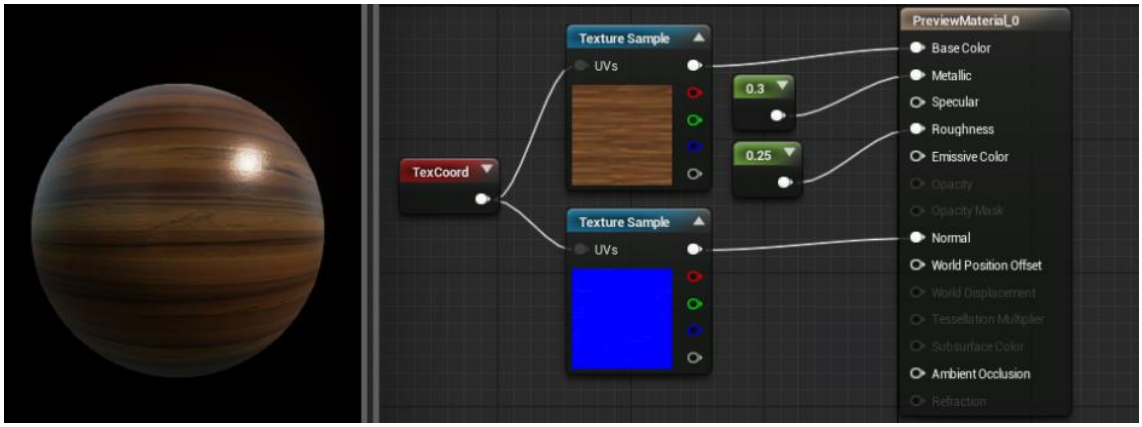

Figura 28.Material que representa un suelo de madera

En este caso, tenemos una red muy simple que define un suelo de madera dura. Sin embargo, estas redes de nodos no necesitan ser tan simples. Es común tener algunos materiales que tienen muchas docenas de nodos dentro de ellos.

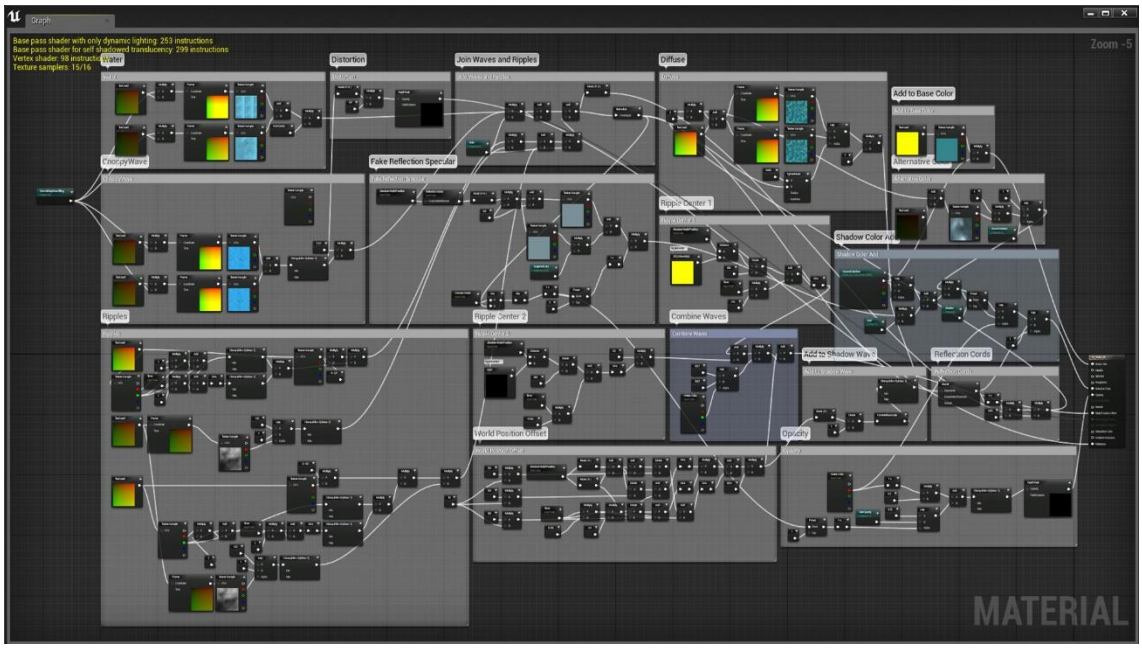

Figura 29. Material que representa el agua de un océano

El color, en términos de imágenes digitales, está dividido en 4 canales principales, los cuales son:

R – Rojo, G – Verde, B – Azul y A – Alfa.

Al igual que con todas las imágenes digitales, para cada píxel, el valor de cualquiera de estos canales puede representarse mediante un número. Gran parte del trabajo de un material es manipular estos números en base a una serie de circunstancias y expresiones matemáticas.

Los materiales usan valores de punto flotante para almacenar datos de color. Esto generalmente significa que los valores para cada canal variarán de 0.0 a 1.0, en lugar de 0 a 255, como lo hacen en algunas aplicaciones de edición de imágenes.

Es importante tener en cuenta que siempre puede saturar los valores, lo que en algunos casos dará lugar a comportamientos especiales. Por ejemplo, si se envía color a la entrada *emissive* de un material, lo que provoca un brillo, los valores superiores a 1,0 aumentarán el brillo del material.

Los nodos que realizan operaciones por canal generalmente necesitan entradas que tengan la misma cantidad de canales. Por ejemplo, se puede agregar un color RGB a otro color RGB, pero no se puede agregar un color RGBA (de cuatro canales) a un color RGB (de tres canales), porque el color RGB no tiene un canal alfa. Esto causa un error y el material no se compilará.

5.3.1.2.1. Entradas

En este apartado veremos algunas de las entradas o *inputs* de los que se compone un material y que más adelante usaremos en nuestro juego a la hora de crear materiales para nuestros modelos.

Se pueden observar en la figura del suelo de madera, cada uno de los pines del material es una entrada. Algunas de estas entradas pueden cambiar ligeramente dependiendo de los cambios que hagamos en los materiales. [29]

5.3.1.2.1.1. *Base Color*

El color base define el color general del material, tomando un Vector3 (RGB) donde cada canal se fija automáticamente entre 0 y 1.

Si se coge del mundo real, este es el color cuando se realiza una fotografía, usando un filtro polarizador (la polarización elimina el espectro de los no metales).

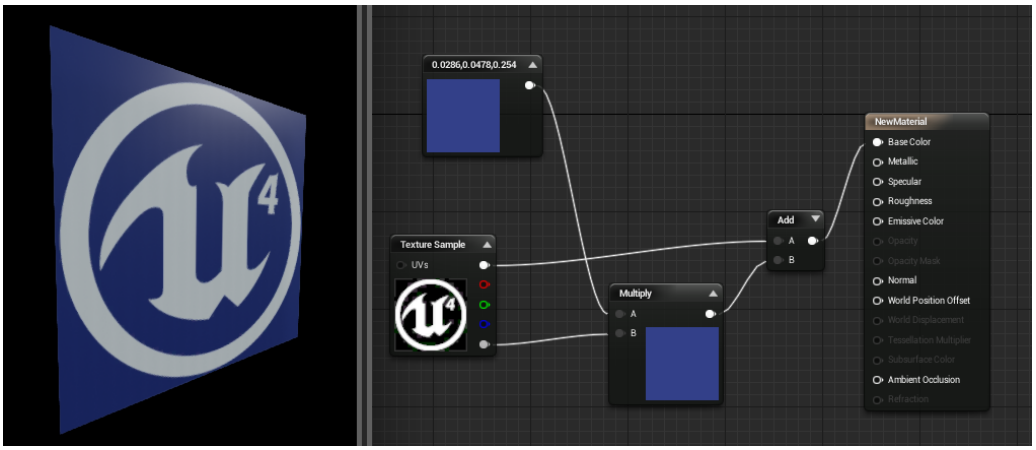

Figura 30. Ejemplo de entrada base color

5.3.1.2.1.2. *Metallic*

La entrada metálica controla cuán "metálica" será la superficie. Los no metales tienen valores metálicos de 0 y los metales tienen valores metálicos de 1. Este valor será 0 o 1 para superficies puras, como metal puro, piedra o plástico.

Al crear superficies híbridas como metales corroídos o polvorientos, es posible que necesite algún valor entre 0 y 1.

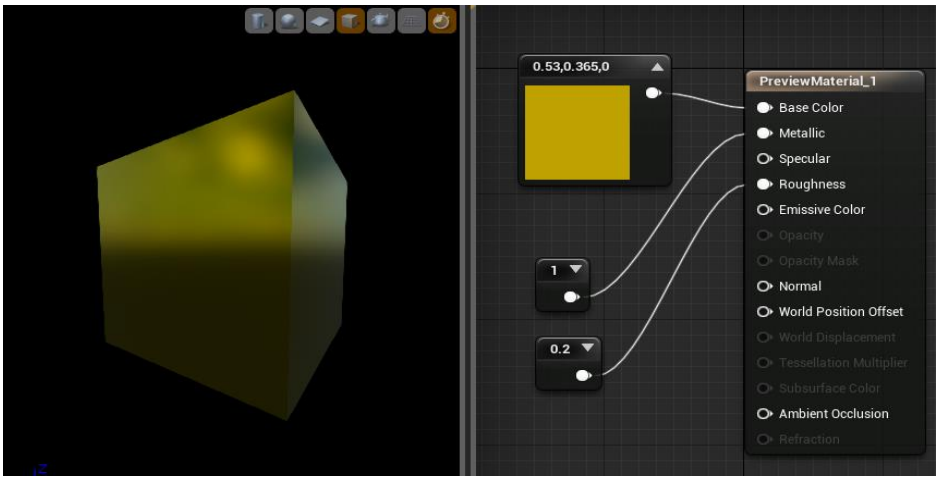

Figura 31. Ejemplo de entrada metallic

5.3.1.2.1.3. *Specular*

Cuando se está editando un material de superficie no metálica, hay momentos en que se querrá ajustar su capacidad para reflejar la luz, específicamente, su propiedad s*pecular*.

Para actualizar la especular de un material, se debe ingresar un valor escalar entre 0 (no reflectante) y 1 (totalmente reflectante). El valor predeterminado de especular de un material es 0.5.

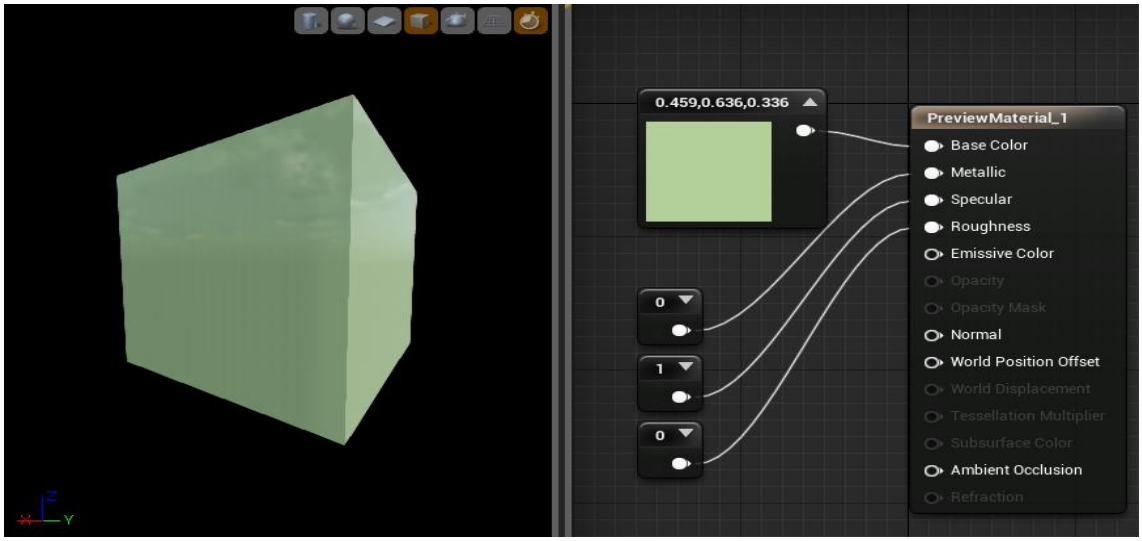

Figura 32. Ejemplo de entrada specular

5.3.1.2.1.4. *Roughness*

La entrada de *Roughness* controla la rugosidad de un Material. Los materiales en bruto dispersan la luz reflejada en más direcciones que los materiales lisos, que es lo difuminado o nítido que es un reflejo. Un valor de rugosidad de 0 (suave) da como resultado una reflexión especular, y un valor de rugosidad de 1 (rugoso) da como resultado una superficie difusa o mate.

La rugosidad es una propiedad que con frecuencia se mapeará en los objetos para agregar la variación más física a la superficie.

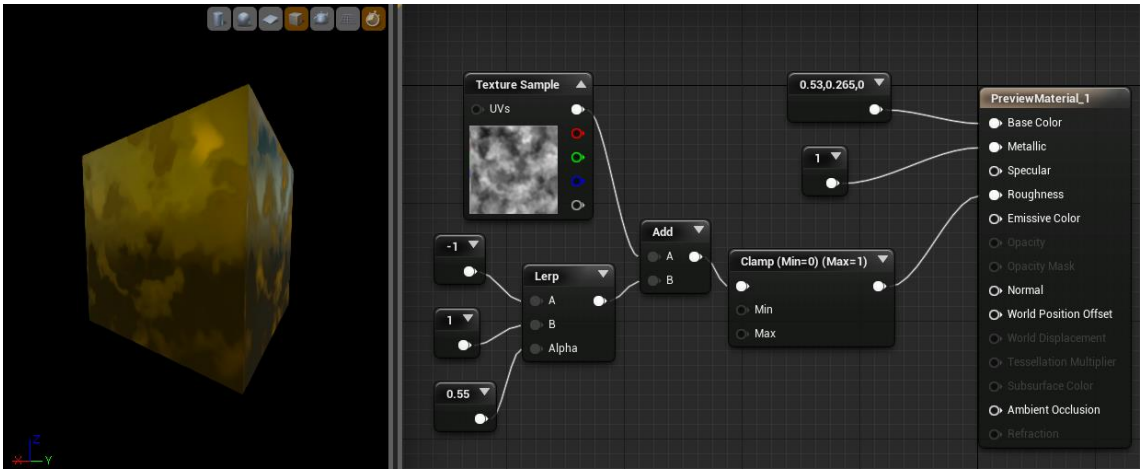

Figura 33. Ejemplo de entrada roughness

5.3.1.2.1.5. *Emissive Color*

La entrada de color emisivo controla qué partes del material parecerán brillar porque están emitiendo luz. Idealmente, se usará una textura enmascarada (principalmente negra, excepto en las áreas que necesitan brillar). Es un recurso muy utilizado para emular el brillo que desprende una bombilla.

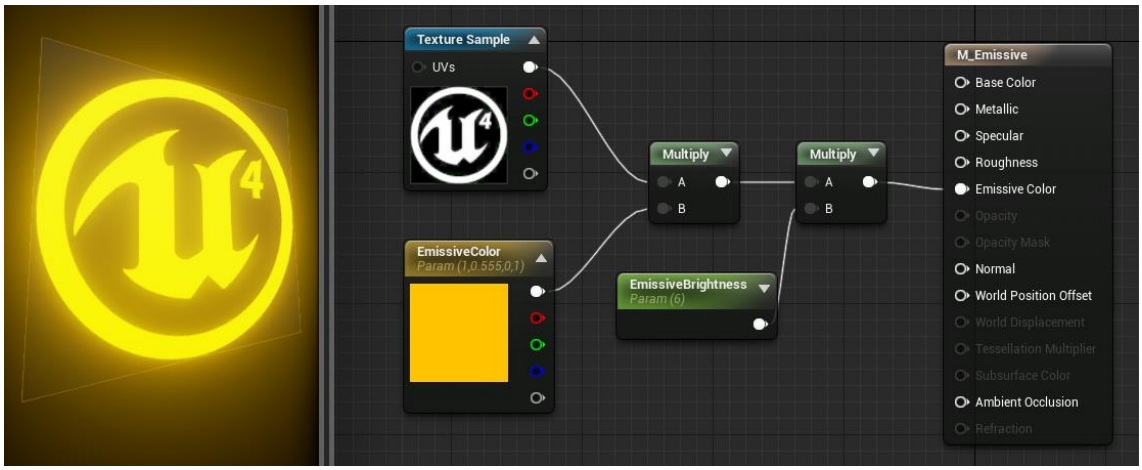

Figura 34. Ejemplo de entrada emissive color

5.3.1.2.1.6. *Opacity*

La entrada de opacidad se usa cuando se usa el modo de mezcla translúcida, una propiedad del material, típicamente para materiales translúcidos, Aditivos y Modulados.

Puede tener un valor entre 0 y 1, donde 0.0 es completamente transparente y 1.0 es completamente opaco.

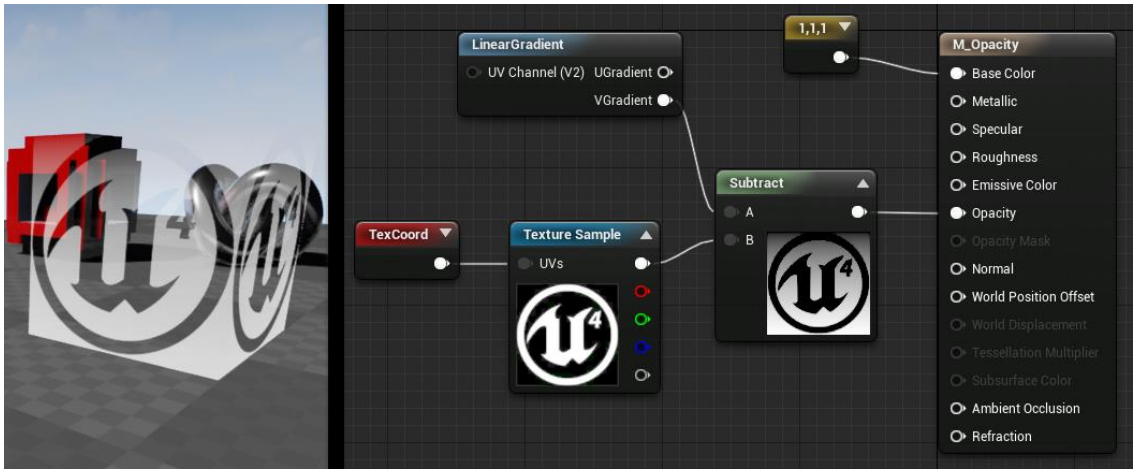

Figura 35. Ejemplo de entrada opacity

5.3.1.2.1.7. *Opacity Mask*

La máscara de opacidad es similar a la opacidad, pero solo está disponible cuando se utiliza el modo de mezcla enmascarada, propiedad de un material. Al igual que con la opacidad, esto toma un valor entre 0.0 y 1.0, pero a diferencia de esta, no se ven distintos tonos de gris en el resultado.

Cuando está en modo enmascarado, un material es completamente visible o completamente invisible. Esto lo convierte en una solución perfecta cuando se necesitan materiales que definen superficies sólidas complejas, como una malla de alambre, cadenas, etc. Las partes opacas seguirán respetando la iluminación.

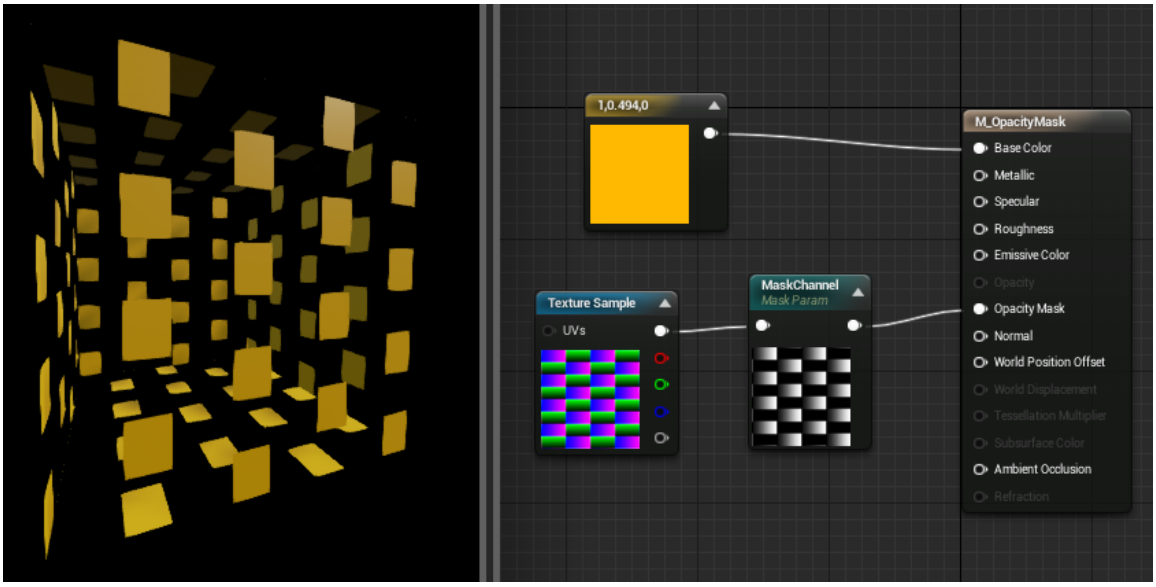

Figura 36. Ejemplo de entrada opacity mask

5.3.1.2.1.8. *Normal*

La entrada normal toma un mapa normal, que se utiliza para proporcionar detalles físicos significativos a la superficie al perturbar la dirección "normal" o de frente de cada píxel individual.

Figura 37. Ejemplo de entrada normal

En la figura de arriba, ambas armas están usando la misma malla estática. La inferior muestra un mapa normal muy detallado, que proporciona detalles adicionales, dando la ilusión de que la superficie contiene muchos más polígonos de los que realmente se representan. Normalmente, los mapas normales se crean a partir de programas de modelado de alta resolución de terceros.

5.3.1.2.1.9. *Subsurface Color*

La entrada de color subsuperficial solo está habilitada cuando la propiedad del modelo de sombreado está configurada en subsuperficial.

Esta entrada le permite agregar un color al material para simular cambios de color cuando la luz pasa a través de la superficie. Por ejemplo, los personajes humanos pueden tener un color rojo subsuperficial en su piel para simular sangre debajo de la superficie.

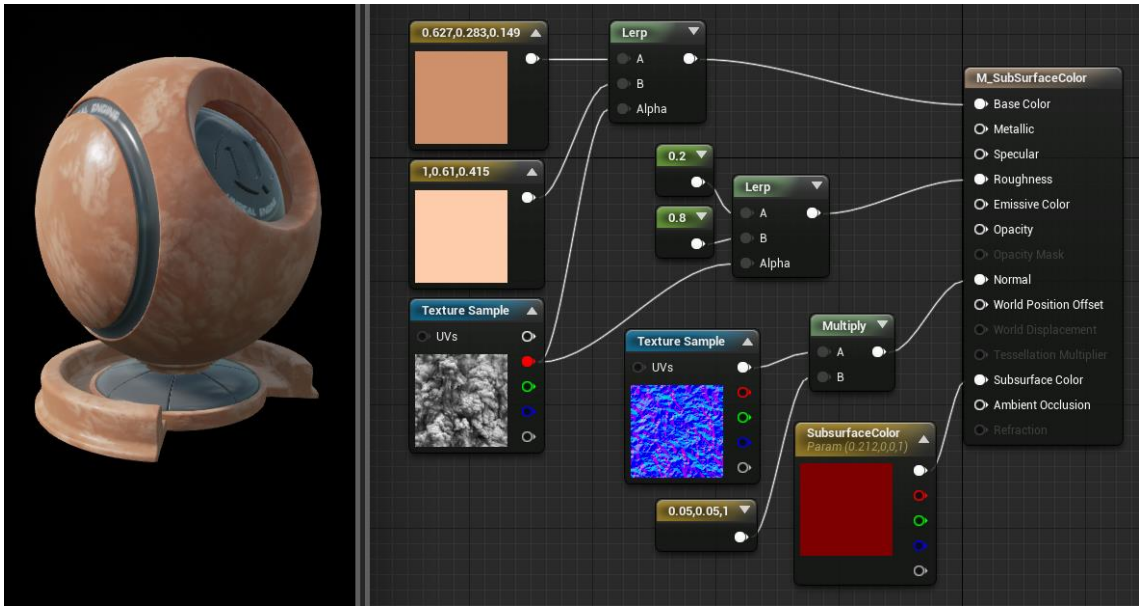

Figura 38. Ejemplo de entrada subsurface color

5.3.1.2.1.10. *Ambient Oclusion*

La oclusión ambiental se usa para ayudar a simular el auto sombreado que ocurre dentro de las grietas de una superficie. En general, esta entrada se conectará a un mapa *Ambient Oclusion* de algún tipo, que a menudo se crea dentro de programas de modelado 3D como *Maya*, *3ds Max* o *Zbrush*, entre otros.

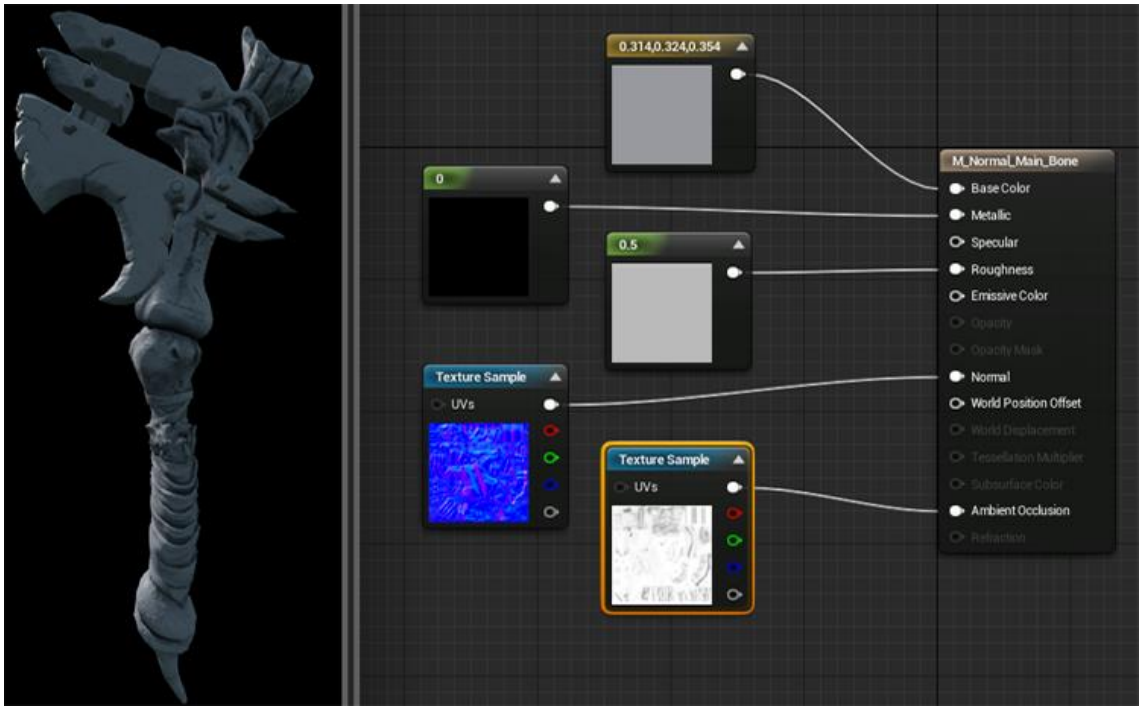

Figura 39. Ejemplo de entrada ambient oclusion

5.3.1.2.1.11. *Refraction*

La entrada de refracción toma una textura o valor que simula el índice de refracción de una superficie. Esto es útil para materiales como el vidrio y el agua, que refractan la luz que pasa a través de ellos.

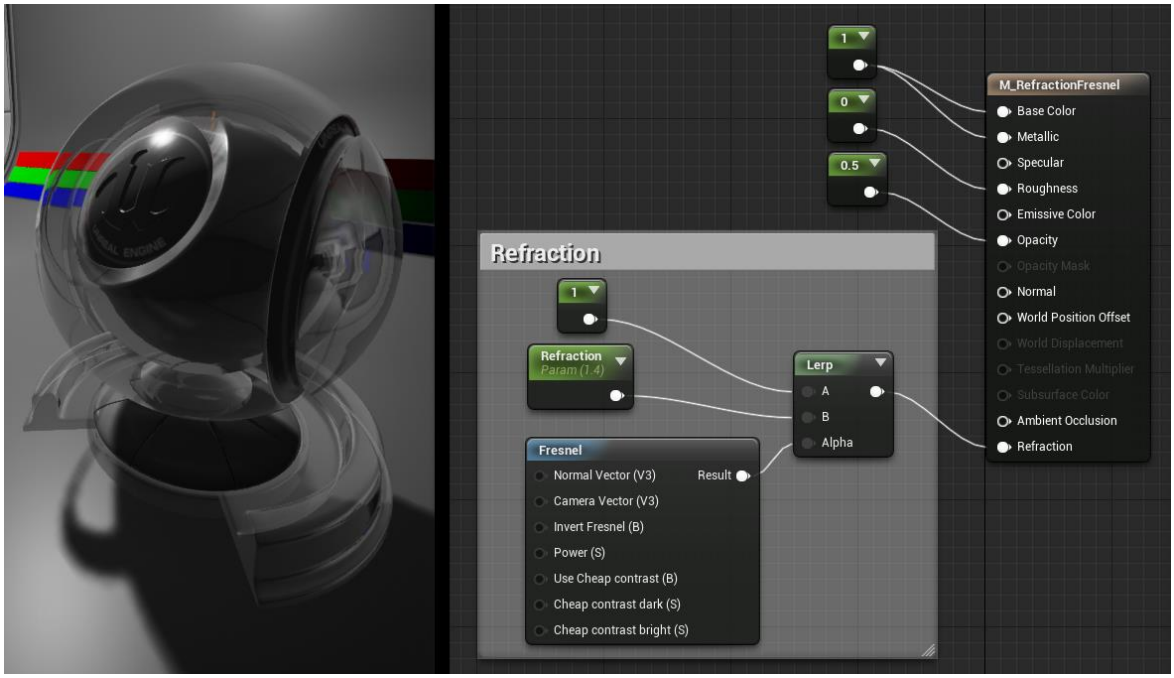

Figura 40. Ejemplo de entrada refraction

5.3.1.4. Texturas

A diferencia de los materiales, las texturas son simplemente imágenes que proporcionan algún tipo de datos basados en píxeles. Estos datos pueden ser del color de un objeto, qué tan brillante es, su transparencia y una variedad de otros aspectos.

Existe un modo de pensar de la vieja escuela de que "texturizar" es cómo aplica color a los modelos de juego. Si bien el proceso de creación de texturas sigue siendo crítico, es importante pensar en las texturas como un componente de los materiales, y no como el resultado final en sí mismas.

Un solo material puede utilizar varias texturas que se muestrean y aplican para diferentes propósitos. Por ejemplo, un material simple puede tener una textura de color base, una textura especular y una textura de mapa normal. Además, puede haber una textura para *emissive* y *roughness* almacenada en los canales *alpha* de una o más de estas texturas. [30]

Una vez creados e importados a *Unreal Editor*, las texturas se incorporan a un material a través de nodos de expresión de materiales especiales, como los nodos de muestra de textura.

Se pueden ver estos en el ejemplo del suelo de madera anteriormente comentado. Estos nodos hacen una referencia al elemento de textura, que existe fuera del material y se puede encontrar por separado en el navegador de contenido.

5.3.2. Creación del nivel

Los modelos de la casa son una mezcla de elementos propios de mi casa y elementos extraídos de diferentes referencias de internet, intentando mantener una misma estética en conjunto.

A continuación, veremos un repaso de los modelados más importantes y el proceso de creación.

5.3.2.1. Paredes

Las paredes del nivel se han realizado de forma modular, de forma que es mucho más sencillo construir el nivel y hacer cambios sobre la marcha. También nos facilita el texturizado ya que conseguimos mayor superficie y mejores detalles.

El modelado de las paredes es un modelado sencillo, inspirado en el diseño de mi propia casa.

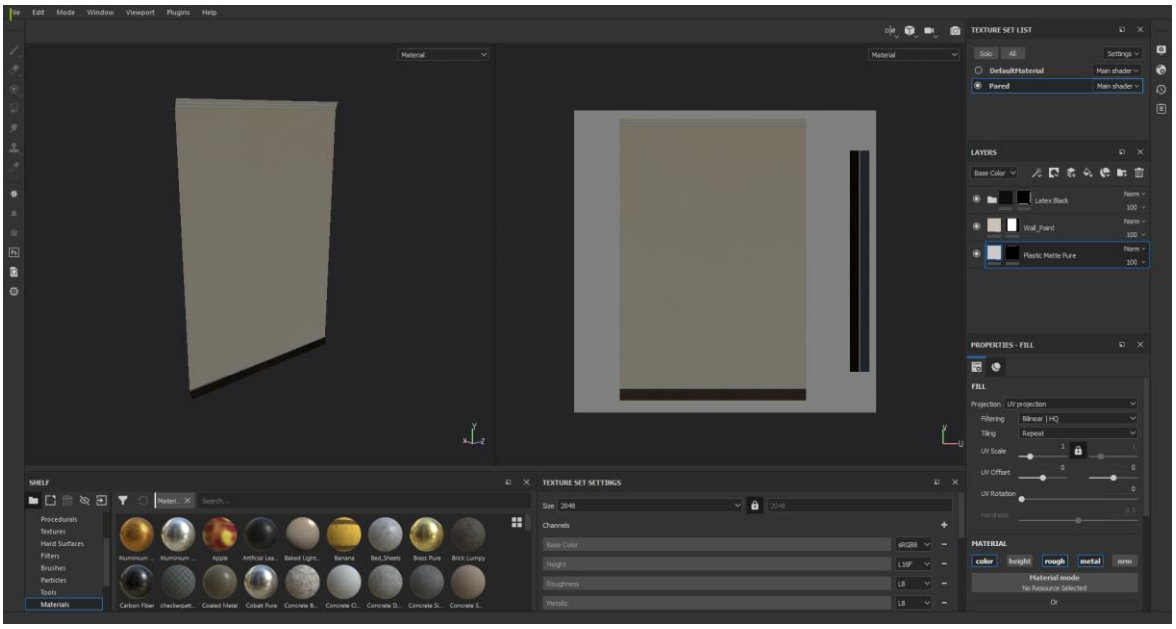

Figura 41. Pared de la casa texturizada en Substance Painter

Debido a la geometría del nivel, se han tenido que realizar diferentes piezas para cubrir todas las necesidades, como esquinas, pilares, paredes con puerta, etc.

Debido a esto, se han tenido que modificar las colisiones para que sean precisas en *Unreal Engine*. Para ello se han de realizar una serie de pasos que veremos en el siguiente apartado.

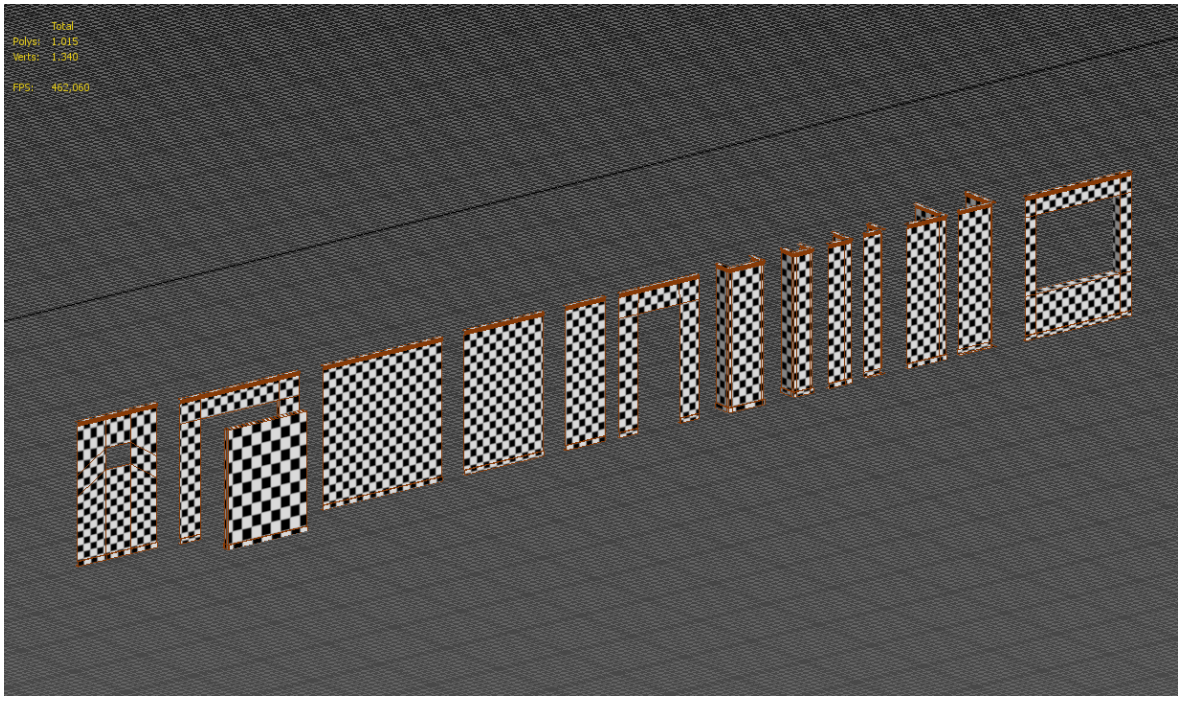

Figura 42. Paredes de la casa en 3ds max con su correspondiente unwrap

Figura 43. Cimientos de la casa construidos

5.3.2.2. Colisiones

Antes de pasar a ver la creación de modelos 3d, tenemos que parar a hablar sobre las colisiones.

Para las colisiones tenemos dos maneras de hacerlas. La primera sería la más simple, importar nuestro modelado a *Unreal Engine* y dejar que el programa ajuste de forma precisa la colisión.

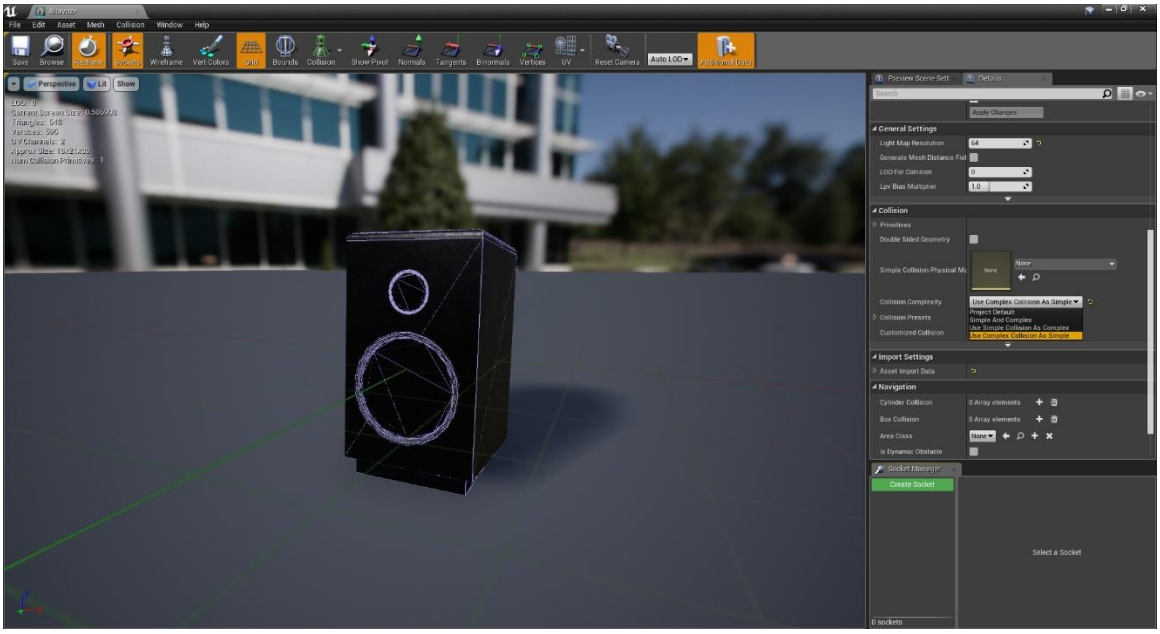

Figura 44. Altavoz con custom collision

La segunda opción sería definir formas básicas que recubran nuestro modelado para hacer una colisión más simple. Esto se consigue realizando un modelado en versión *low poly*, recubriendo nuestro modelado original.

Vamos a ver un ejemplo con el modelo del sofá. Una vez tenemos el modelo realizado en el programa de modelado, tenemos que recubrirlo con formas muy básicas.

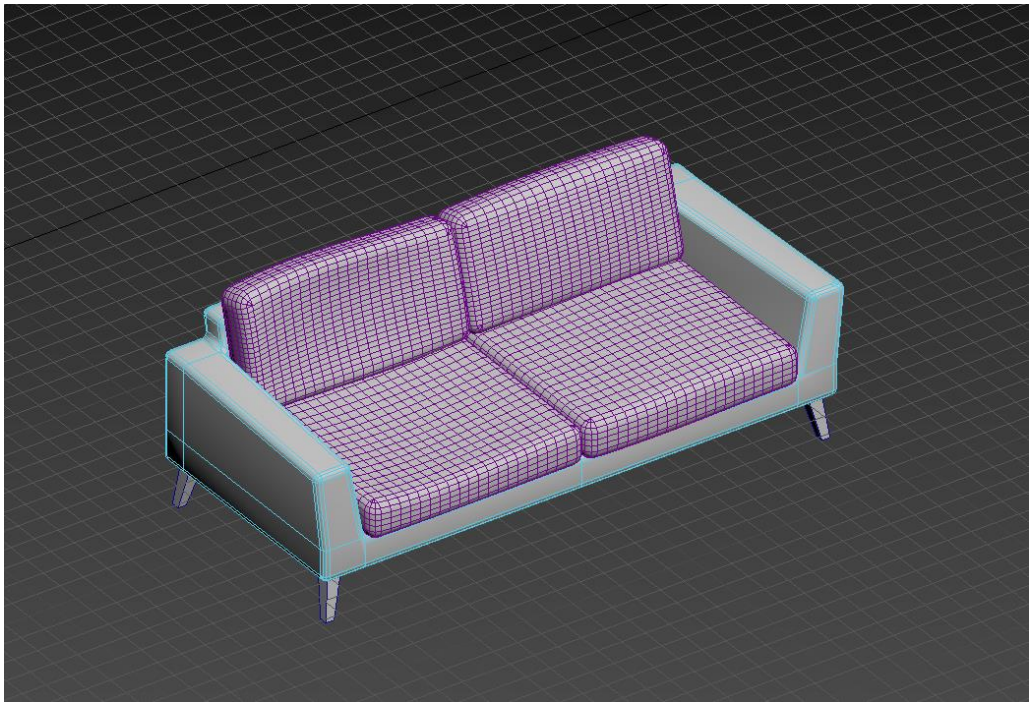

Figura 45. Modelo del sofá del comedor

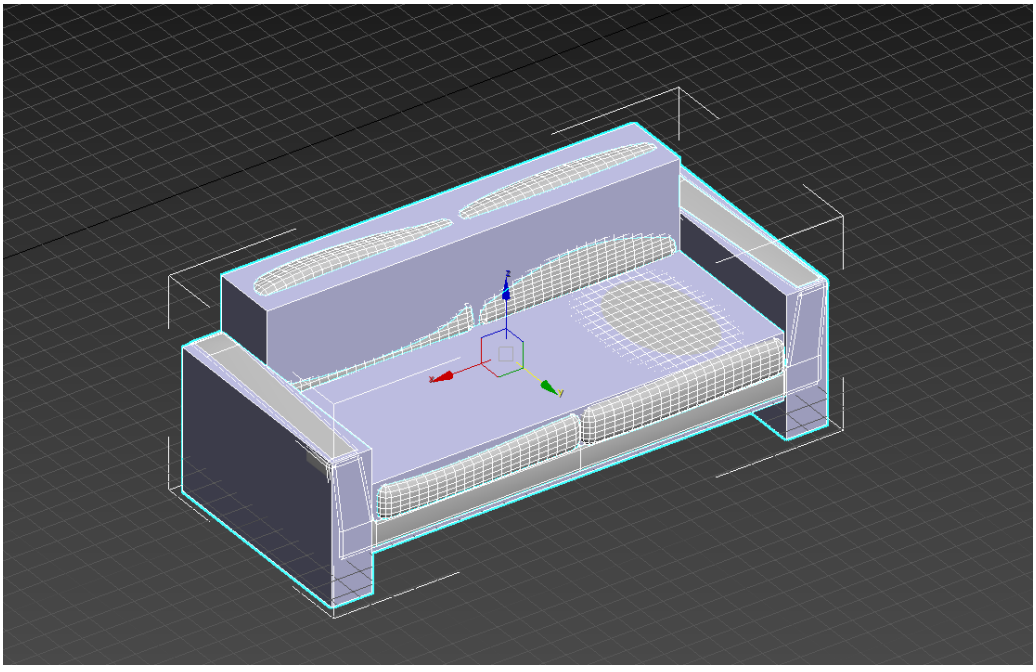

Figura 46. Creación de colisiones customizadas para el sofá

A continuación, definimos el nombre, muy importante, debe seguir la siguiente nomenclatura "UCX_nombreObjeto_0X". Una vez estén todas con el nombre correcto, se agrupan en *3ds max* y se exporta como fbx.

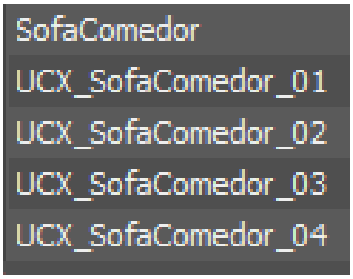

Figura 47. Nomenclatura para la correcta detección de la colisión personalizada

De esta forma al importar nuestro nuevo objeto, unreal reconoce las colisiones, y a la hora de definirlas utiliza la versión *low poly* que se había realizado.

En nuestro caso se han utilizado ambos tipos de colisiones. Dependiendo de la complejidad del modelo, podemos utilizar la colisión compleja en aquellos casos que no se utilicen muchos polígonos y la colisión personalizada en otros casos donde el modelo tenga muchos polígonos.

La colisión compleja es mucho más rápida, pero consume mucho más, ya que tiene que calcular la colisión por cada uno de los vértices, lo que ocasiona pérdida de rendimiento.

5.3.2.3. Modelos 3D

En este apartado veremos el proceso de creación de los modelos 3d de los que se compone el nivel.

Como se ha comentado a lo largo de este documento, todos los modelos 3d de los que se compone el juego, han sido realizados desde cero, sin utilizar modelos externos.

Se han realizado diferentes técnicas a la hora de modelar y se han cuidado todos los mapas UV para que el resultado final en *Unreal Engine* sea el esperado. También se ha intentado conseguir el mayor nivel de detalle utilizando el menor número de polígonos posible, de forma que el juego final no tenga problemas de rendimiento.

Cabe destacar que algunos modelos se han reusado en diferentes partes de la casa, aunque con ligeras variaciones, para tener mayor variedad de modelos.

Debido a la gran cantidad de modelos 3d de los que se compone la casa, veremos los modelos resumidamente zona por zona:

5.3.2.3.1. Dormitorio 1

En el dormitorio 1 podemos encontrar diferentes objetos que en su conjunto representan la habitación de una adolescente.

En la foto de abajo podemos ver los modelos de los que se compone la habitación, junto a los polígonos y vértices que contiene.

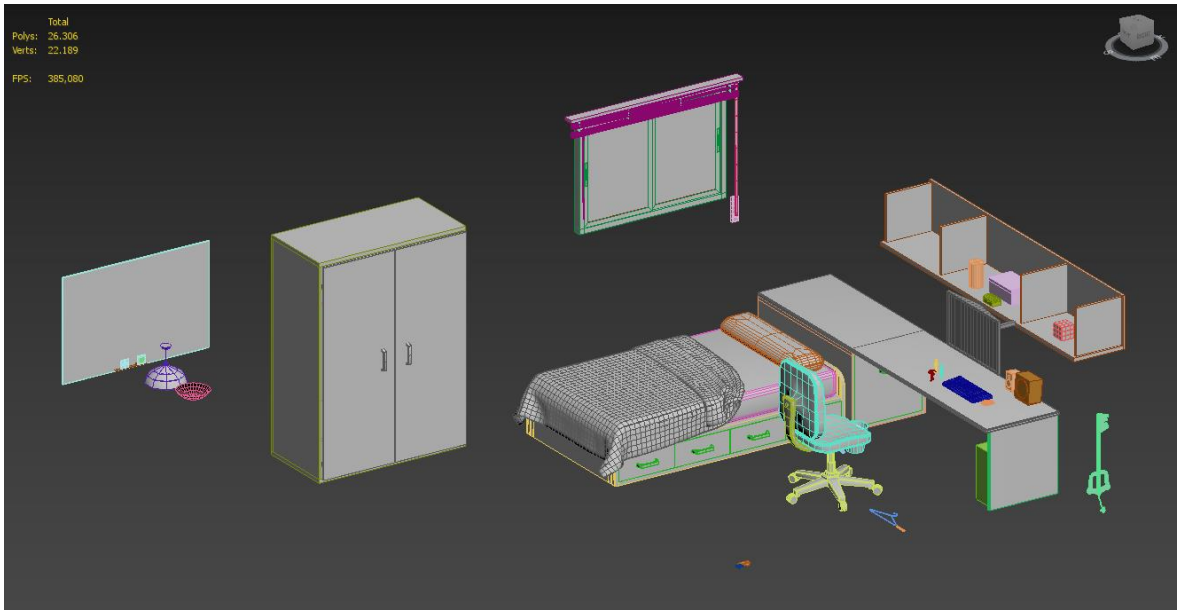

Figura 48. Modelos 3d del dormitorio 1

Aproximadamente el dormitorio 1 cuenta con 26.000 polígonos y 22.000 vértices entre todos los modelos.

5.3.2.3.2. Dormitorio 2

En el dormitorio 2 podemos encontrar diferentes objetos que en su conjunto representan la habitación de matrimonio. Cuenta con unos detalles religiosos, dándonos a entender que es de un matrimonio católico.

En la foto de abajo podemos ver los modelos de los que se compone la habitación, junto a los polígonos y vértices que contienen.

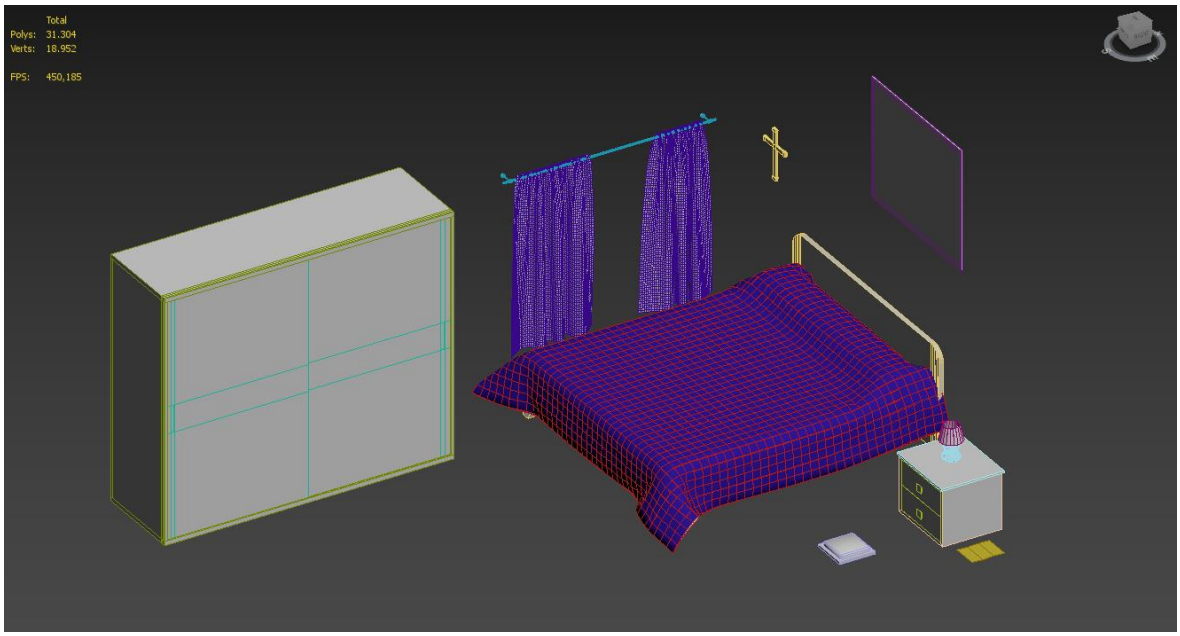

Figura 49. Modelos 3d del dormitorio 2

Aproximadamente el dormitorio 2 cuenta con 31.000 polígonos y 19.000 vértices entre todos los modelos.

5.3.2.3.3. Aseo

En el aseo podemos encontrar diferentes objetos que en su conjunto representan un aseo familiar. Cuenta con unos detalles visuales como los cepillos de dientes o las esponjas, dándonos a entender que vive una familia en la casa.

En la foto de abajo podemos ver los modelos de los que se compone el aseo, junto a los polígonos y vértices que contiene.

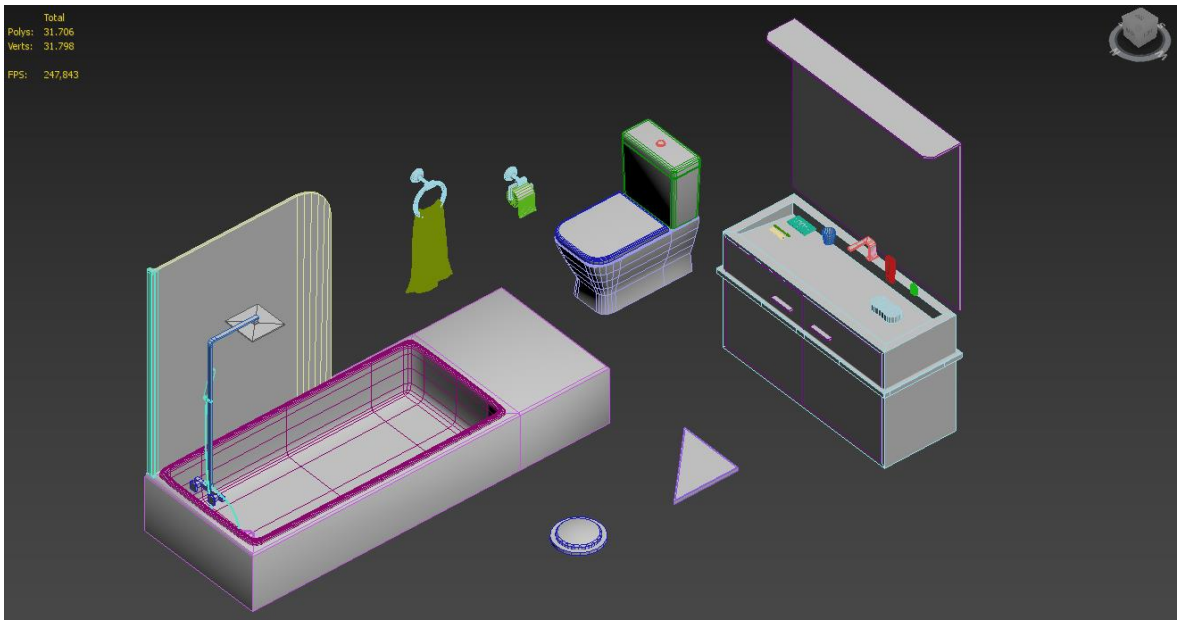

Figura 50. Modelos 3d del aseo

Aproximadamente el aseo cuenta con 31.000 polígonos y 31.000 vértices entre todos los modelos.

5.3.2.3.4. Cocina

En la cocina podemos encontrar diferentes objetos representando una cocina simple, pero que cuenta con todo lo necesario. Se pueden encontrar algunos restos de comida y platos sucios, que representan que recientemente vivía alguien en la casa.

En la foto de abajo podemos ver los modelos de los que se compone la cocina, junto a los polígonos y vértices que contiene.

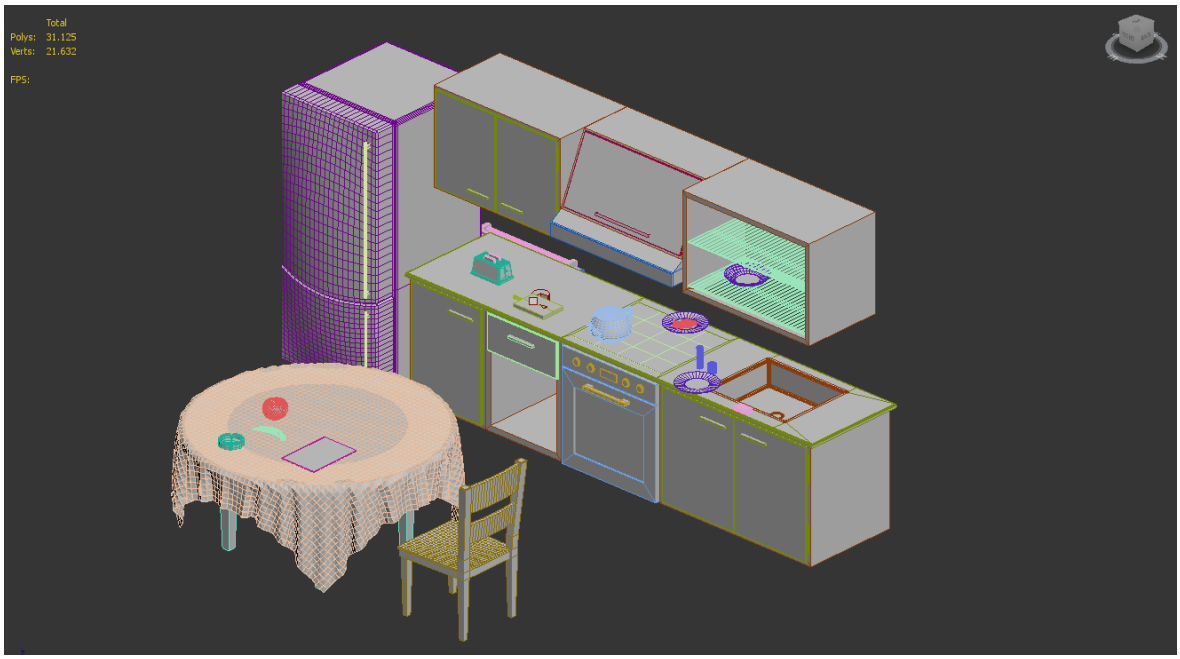

Figura 51. Modelos 3d de la cocina

Aproximadamente la cocina cuenta con 31.000 polígonos y 21.000 vértices entre todos los modelos.

5.3.2.3.5. Comedor

En el comedor podemos encontrar diferentes objetos representando un comedor y sala de estar, contando con todo lo necesario.

En la foto de abajo podemos ver los modelos de los que se compone el comedor, junto a los polígonos y vértices que contiene.

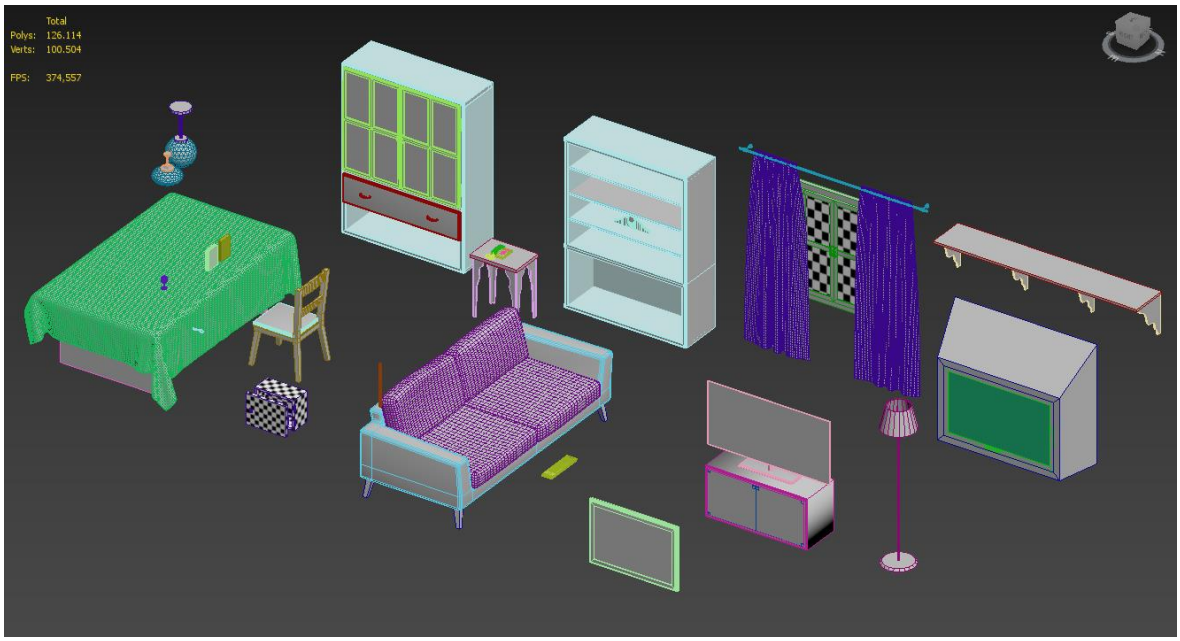

Figura 52. Modelos 3d del salón

Aproximadamente el comedor cuenta con 126.000 polígonos y 100.000 vértices entre todos los modelos.

5.3.2.3.6. Pasillo

En el pasillo podemos encontrar diferentes objetos, aunque muchos se comparten con el comedor.

En la foto de abajo podemos ver los modelos de los que se compone el pasillo, junto a los polígonos y vértices que contiene.

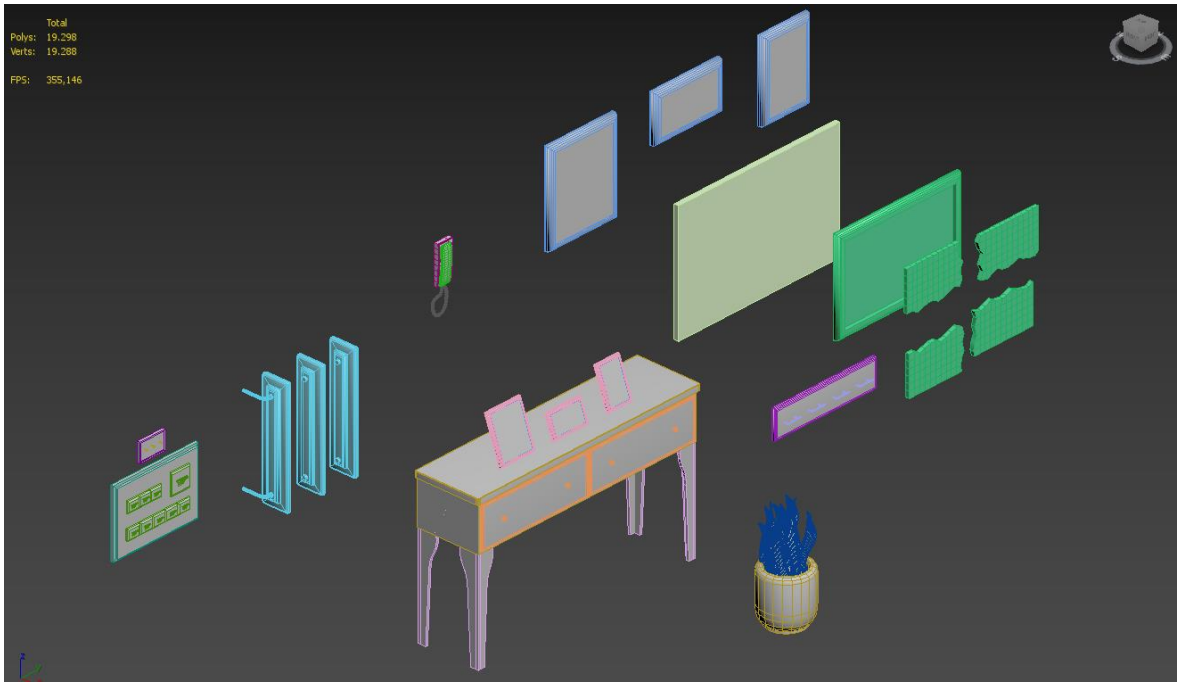

Figura 53. Modelos 3d del pasillo

Aproximadamente el pasillo cuenta con 19.000 polígonos y 19.000 vértices entre todos los modelos.

Cabe destacar que el radiador se ha realizado de forma modular, es decir, que solo con 3 piezas podemos obtener diferentes radiadores de diferente longitud, haciendo copias de la parte central.

5.3.2.3.7. Galería

En la galería podemos encontrar diferentes objetos de uso cotidiano.

En la foto de abajo podemos ver los modelos de los que se compone la galería, junto a los polígonos y vértices que contiene.

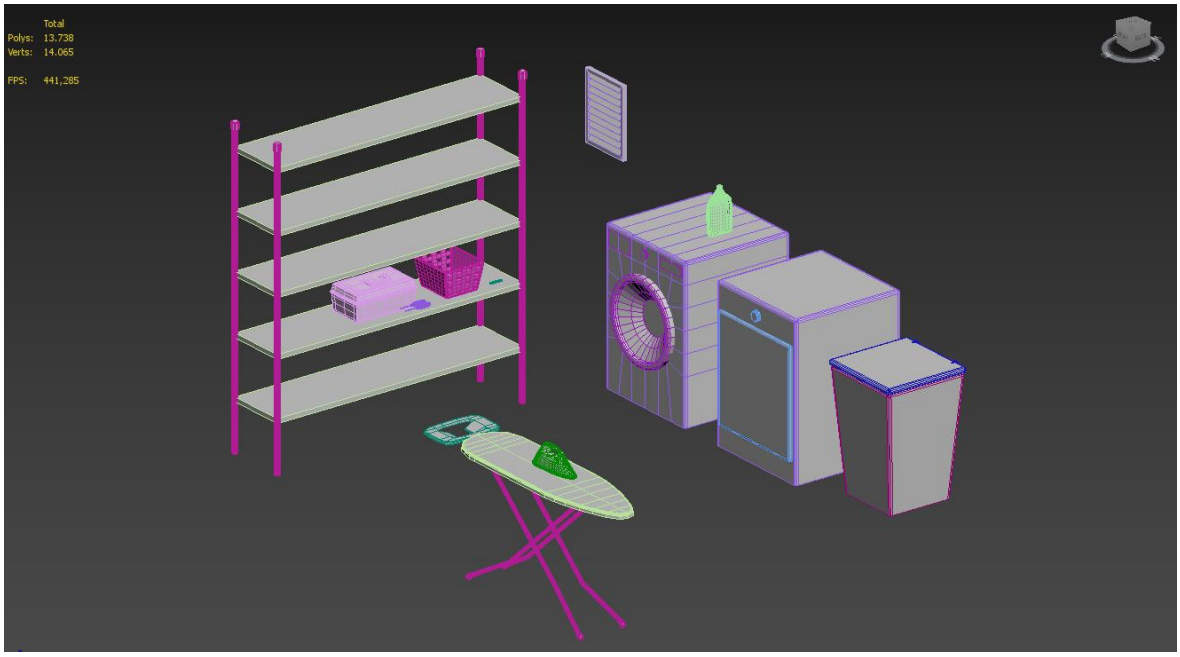

Figura 54. Modelos 3d de la galería

Aproximadamente la galería cuenta con 14.000 polígonos y 14.000 vértices entre todos los modelos.

5.3.2.3.8. Habitación secreta

En la habitación secreta podemos encontrar diferentes objetos que reflejan que alguien ha sido encerrado en ella.

En la foto de abajo podemos ver los modelos de los que se compone la habitación secreta, junto a los polígonos y vértices que contiene.

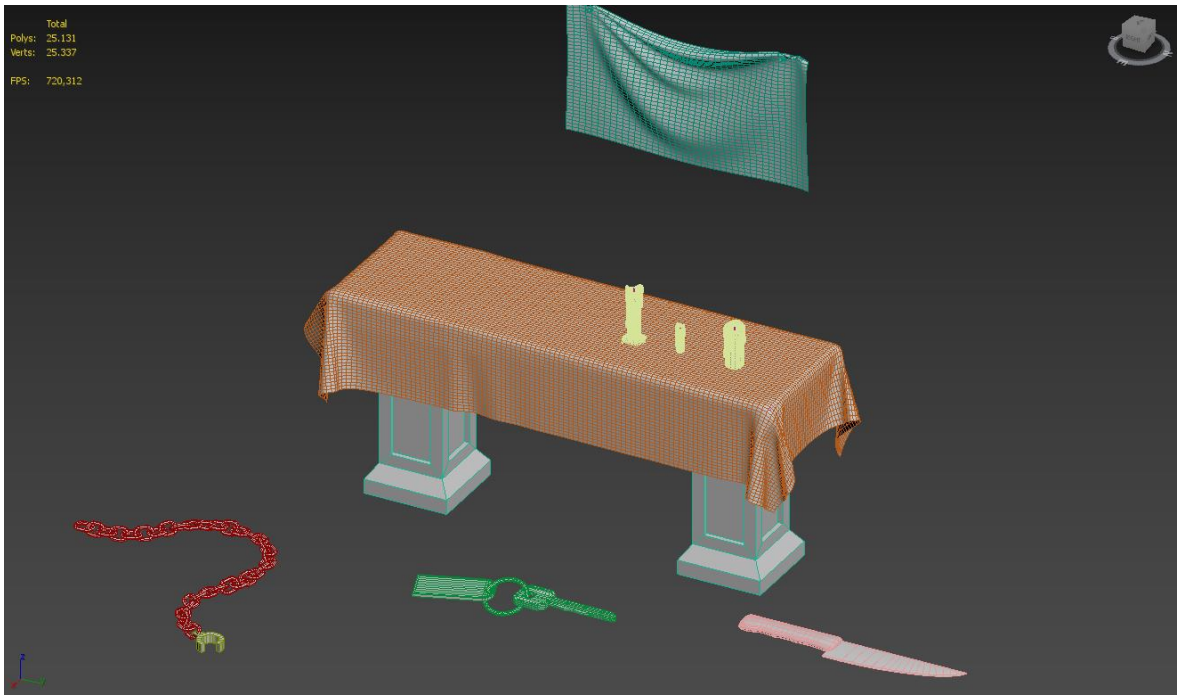

Figura 55. Modelos 3d de la habitación secreta

Aproximadamente la habitación secreta cuenta con 25.000 polígonos y 25.000 vértices entre todos los modelos.

5.3.2.3.9. Recuento final

Una vez comentados todos los modelos del nivel, podemos observar que aproximadamente contamos con 330.000 polígonos y 270.000 vértices en todo el juego. Esto nos ofrece un muy buen rendimiento y se ha conseguido crear un nivel muy detallado con una cantidad controlada de polígonos y vértices.

5.3.2.4. Texturizado

En este apartado veremos el texturizado de algunos de los modelos que se pueden encontrar en el nivel.

Antes de texturizar un modelo, tenemos que asegurarnos que su *unwrap* este bien realizado, ya que si no lo hacemos bien puede dar problemas a la hora de verse la textura, así como problemas a la hora de recibir luz.

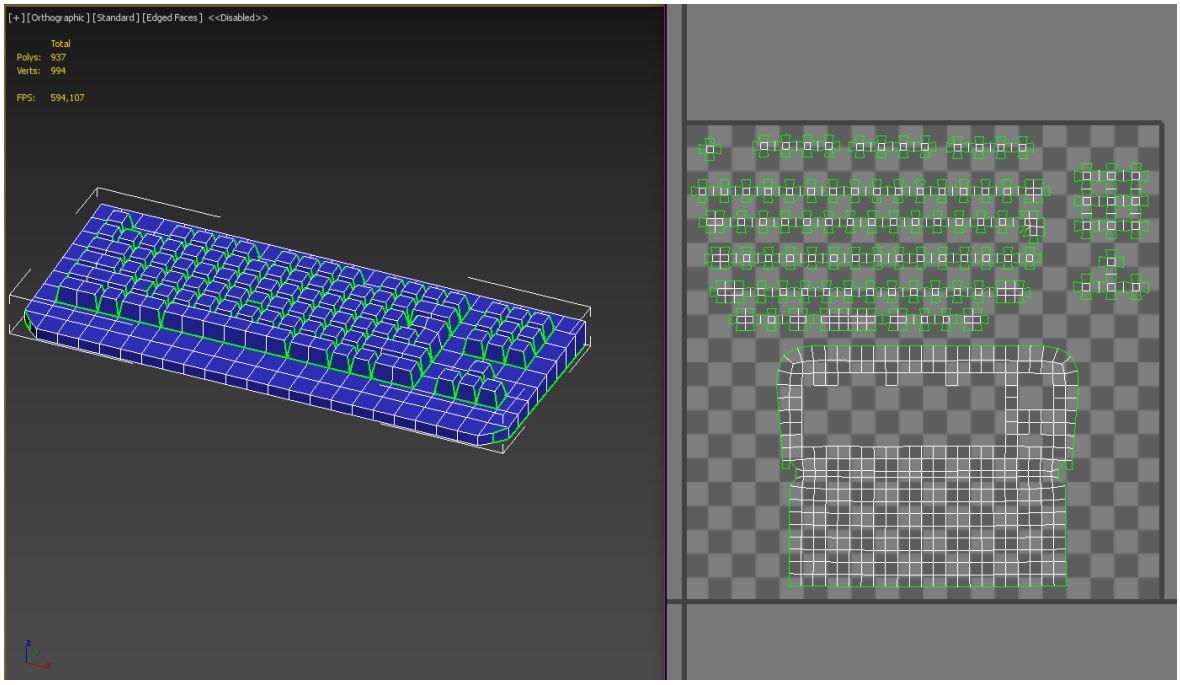

Figura 56. Unwrap del teclado del dormitorio 1

Con Substance Painter es muy sencillo hacer texturas básicas, ya que solo debemos importar el modelo y aplicarle un material, pero para este proyecto se han llevado a cabo diferentes técnicas y uso de máscaras para obtener grandes niveles de detalle.

Veamos paso a paso como se ha realizado el texturizado de algunos de los modelos del juego.

Una vez importado nuestro modelo a *Substance Painter*, podremos ver la interfaz principal del programa. Lo primero que podemos observar son dos ventanas, una en la que podemos ver el modelo, rotarlo, acercarlo, etc., y otra donde podemos ver su mapa UV.

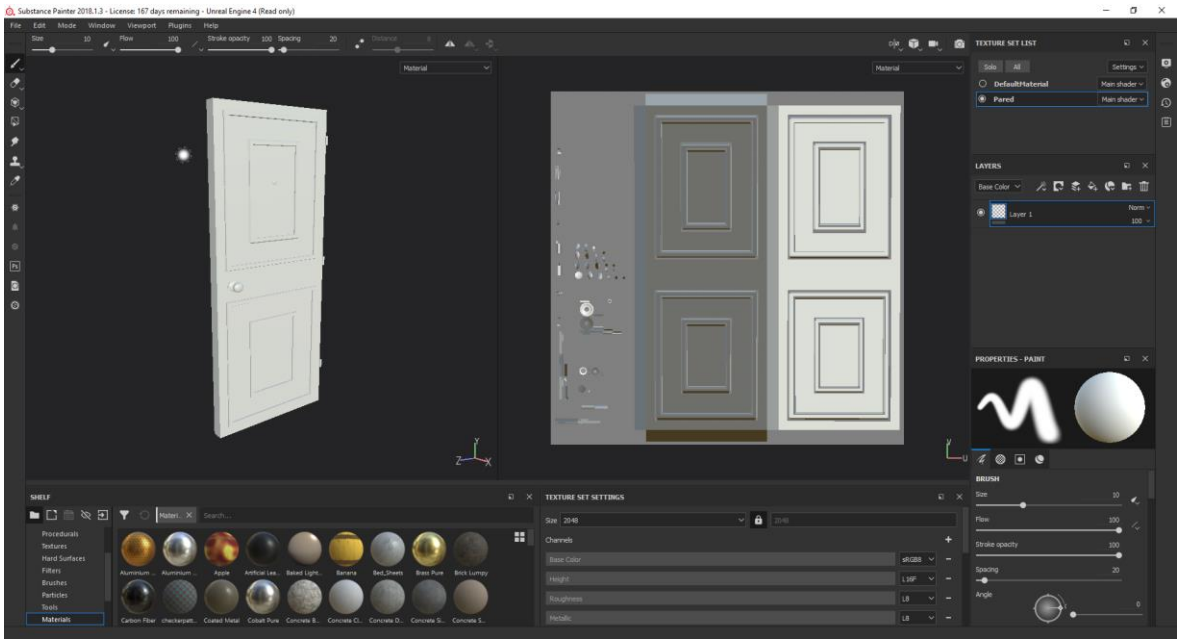

Figura 57. Interfaz de Substance Painter con el modelo de la puerta principal

En otra ventana tenemos toda la lista de materiales y herramientas que podremos usar para obtener diferentes resultados. Desde la página *Substance Share*, propiedad de *Allegorithmic*, podemos descargar miles de materiales y herramientas creados por la comunidad de usuarios de forma gratuita y sin derechos de uso.

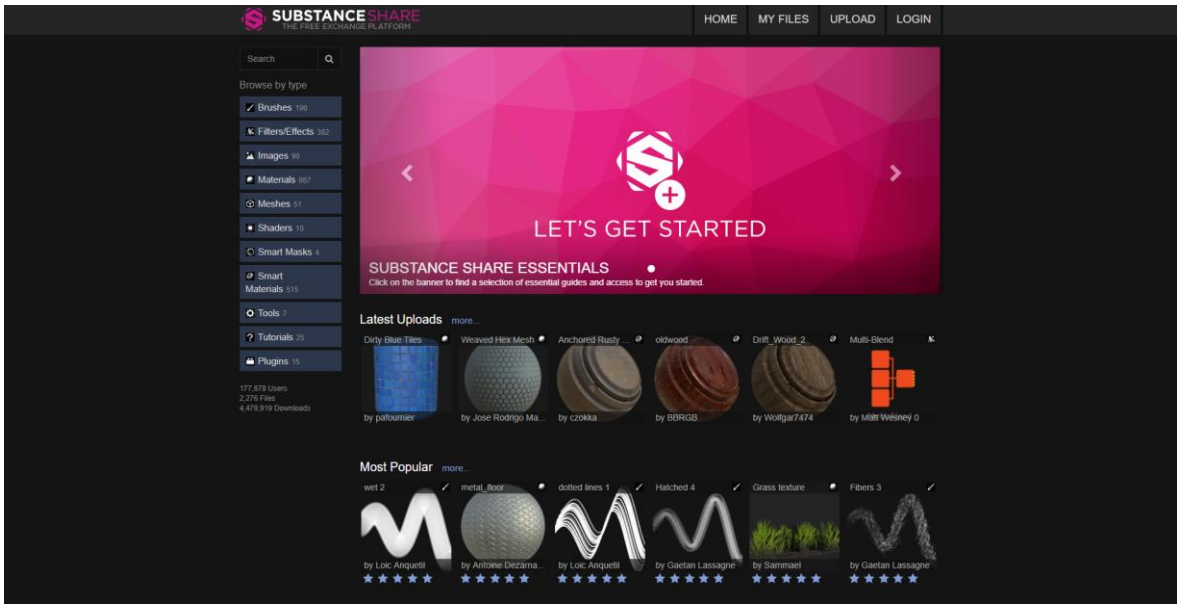

Figura 58. Página substance share

Una vez elegidos los materiales, podemos trabajar de una forma muy parecida a *Photoshop*, ya que podemos usar diferentes capas y mascaras para trabajar sobre zonas concretas del modelo.

En el caso de la puerta que estamos tomando como ejemplo, se ha optado por un tipo determinado de madera, junto a diferentes metales para las bisagras y los pomos de las puertas.

También se han hecho detalles como los tornillos y algunas marcas de desgaste sobre el modelo. Esto ofrece un gran nivel de detalle para el jugador, que podrá ver ciertos detalles durante la exploración.

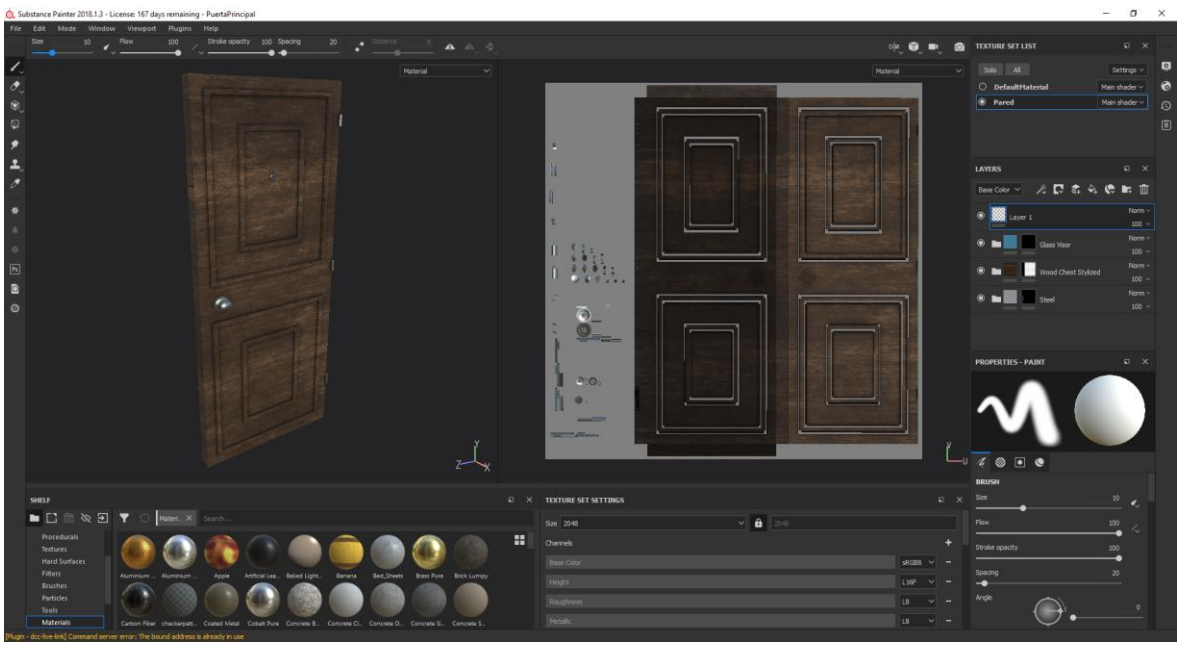

Figura 59. Puerta principal texturizada

Tenemos que exportar las texturas para poder utilizarlas en el motor, para ello Substance Painter tiene un tipo de configuración que nos facilita la tarea, exportando las tres texturas que necesitamos en *Unreal Engine*.

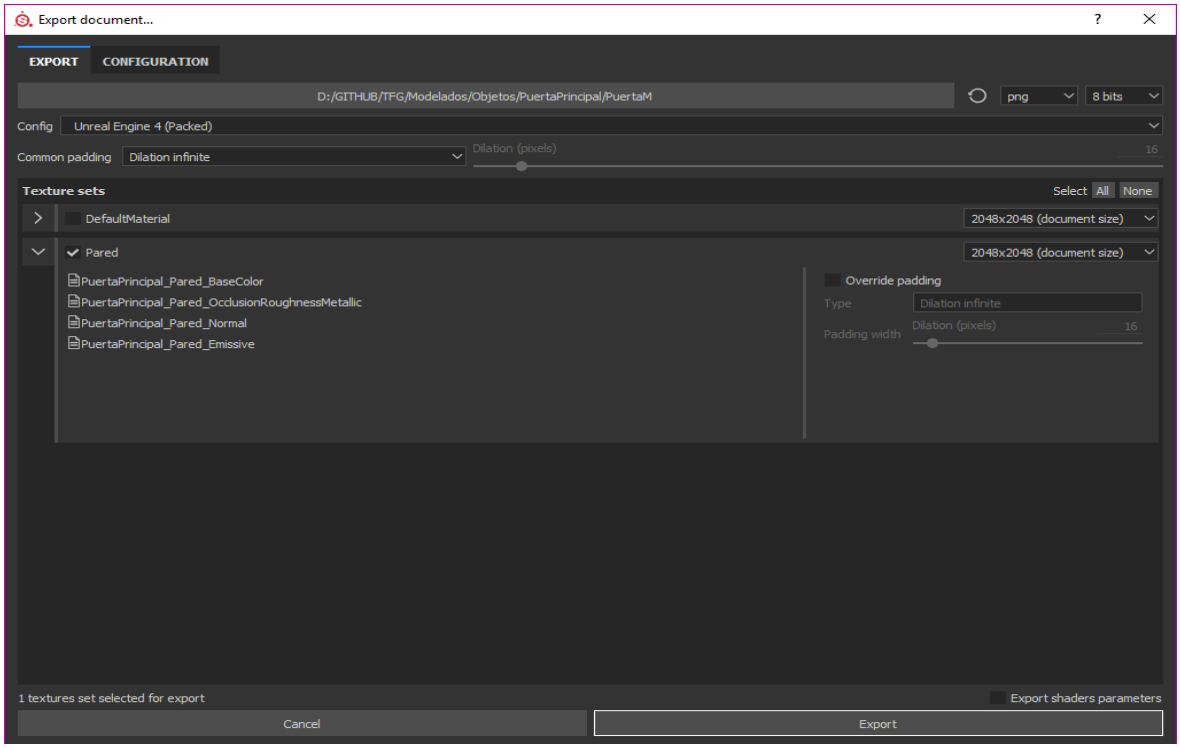

Figura 60. Exportación de texturas

Por último, destacar que para algunos modelos se ha realizado un texturizado complementario en *Photoshop*, sobre las texturas generadas en *Substance Painter* para resaltar ciertos detalles que no se podían realizar desde *Substance Painter*.

Relacionado a esto último, todas las imágenes utilizadas en este proyecto se han recopilado de *Pixabay*, *Textures.com* y *Pngimg.com*.

Pixabay es una web donde tenemos muchísimas imágenes sin derechos de *copyright*, por lo que viene genial para cuadros, libros, etc. *Textures.com* es una web donde se pueden conseguir cientos de texturas y efectos sin derechos de uso, al igual que *Pngimg.com*.

Estas dos últimas se han utilizado para el uso de *decals*, que son unos materiales especiales que se utilizan para simular manchas o suciedad.

5.3.2.5. Materiales

Una vez texturizados los modelos 3d de nuestro juego, tenemos que importarlos a *Unreal Engine* junto con las texturas extraídas de *Substance Painter*.

Si previamente hemos definido el material en *3ds max*, cuando importamos el modelo 3d, *Unreal Engine* nos crea automáticamente un material asociado a la malla, con el mismo nombre que se le dio en *3ds max*. En el caso que no se haya asociado un material previamente, podemos crear uno nuevo y asociarlo al modelo 3d directamente desde *Unreal*.

Como se ha visto anteriormente, cuando exportamos las texturas de *Substance Painter*, se nos generan 3 archivos por cada material del que se componga el modelo 3d.

Figura 61. Texturas para la puerta principal

Con estos 3 archivos, una vez importados a unreal, debemos rellenar el material que previamente habíamos creado. Primero tenemos que desmarcar la casilla de sRGB y dejarla en false para que luego no tengamos problemas de visualización ya que usaremos los 3 canales RGB de la textura por separado. Si no hacemos esto, tendremos problemas con algunos reflejos del metal ya que *Unreal Engine* toma esa textura como un único canal.

Posteriormente tenemos que conectar las 3 imágenes al material de la manera que se muestra en la siguiente figura.

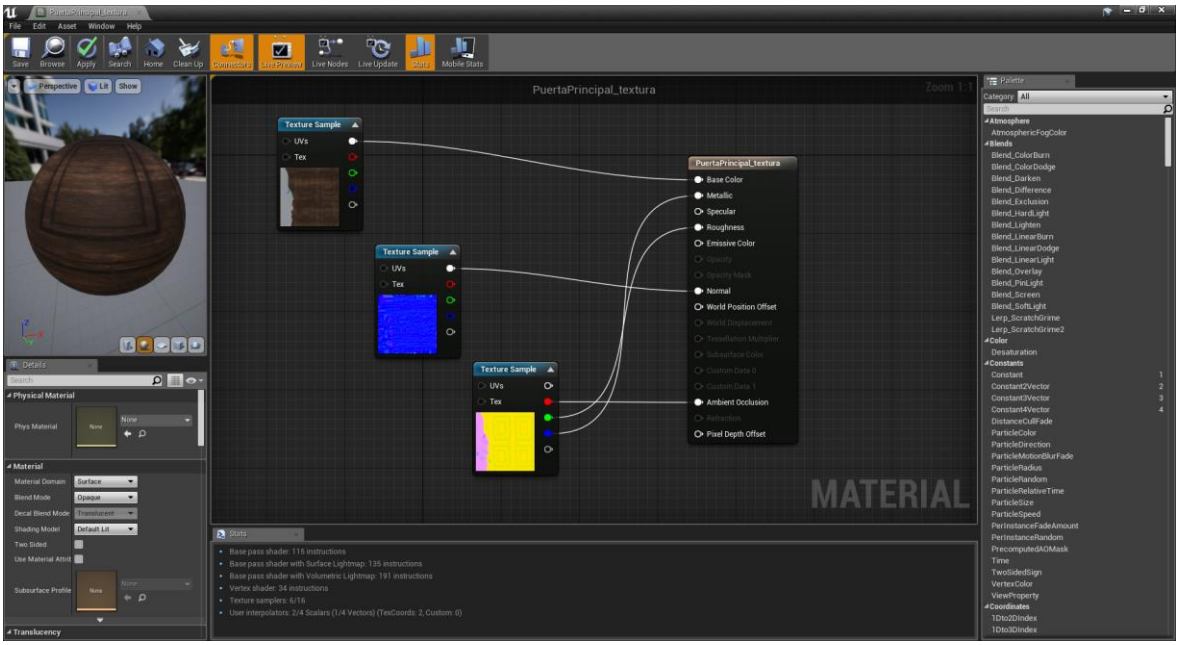

Figura 62. Creación del material de la puerta principal con las texturas anteriores

Previamente en el apartado de 5.3.1.3. Materiales, ya describimos cada una de las entradas de un material, por lo que no entraremos en demasiados detalles. Simplemente observar que, cada una de las imágenes anteriormente generadas en *Substance Painter*, se conecta con los nodos correspondientes, obteniendo un material completo.

5.3.3. Desarrollo de mecánicas

En este apartado vamos a ver en más detalle las mecánicas anteriormente comentadas en el apartado 5.2.3. Mecánicas. También vamos a ver algunos de los eventos que se pueden ver durante el juego.

Cabe destacar que no se va a entrar en detalle de las diferentes clases de objetos que tiene *Unreal Engine*, como los actores o herencias, ya que no es el objetivo de este proyecto.

5.3.3.1. Movimiento y cámara

Lo primero que tenemos que realizar es la configuración de los *inputs* para el juego. Para ello en los ajustes del proyecto en *Unreal Engine*, en la pestaña *input*, debemos definir el movimiento y la vista en los dos ejes, x e y.

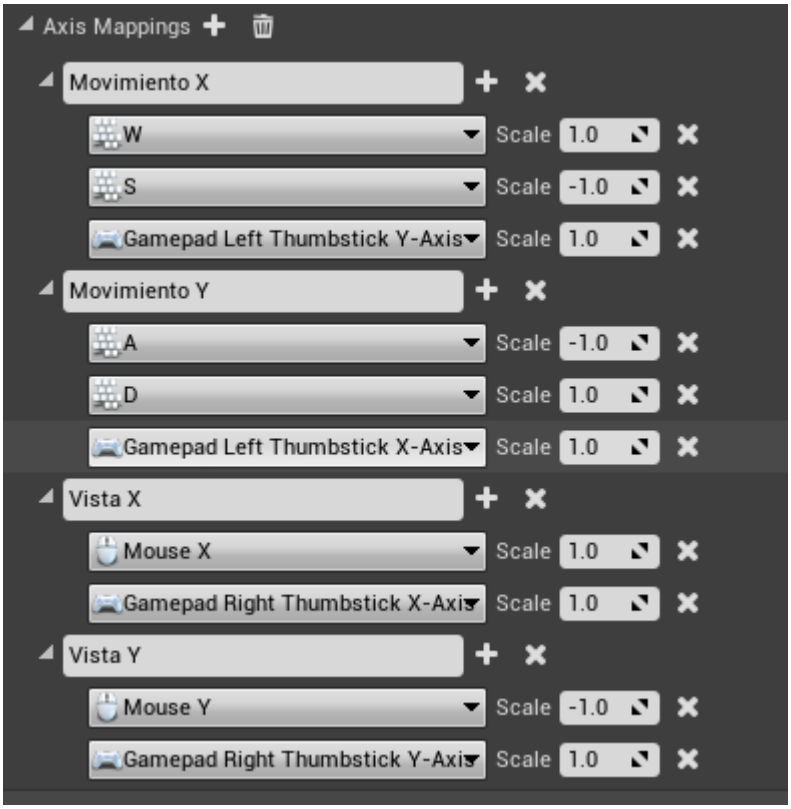

Figura 63. Mapeo de movimiento y cámara

Una vez realizado esto, podemos referenciarlos en el *blueprint* del jugador y obtener los valores de entrada del movimiento y la cámara.

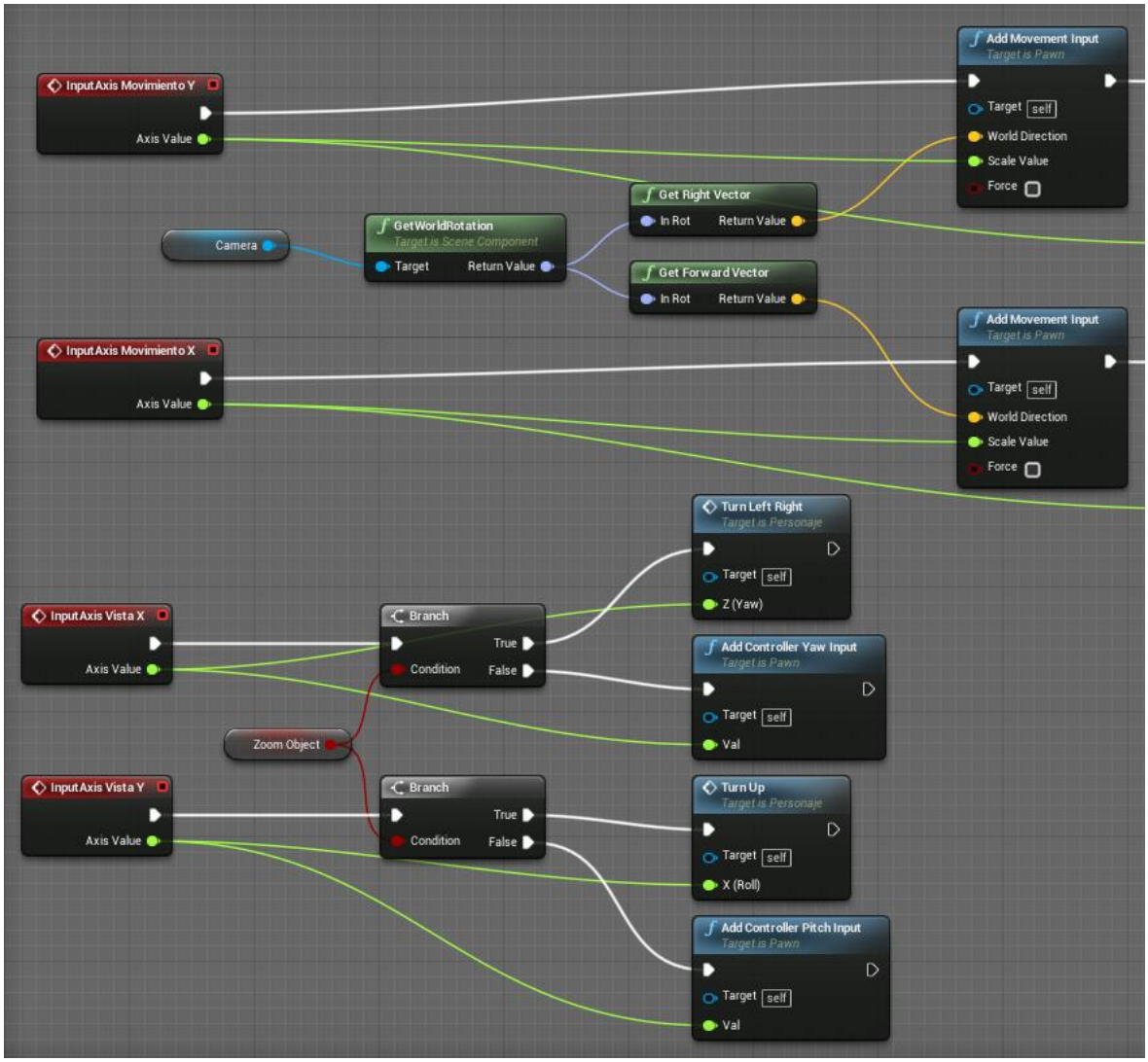

Figura 64. Blueprints movimiento y cámara

Con esto, ya tendríamos el movimiento y la cámara base. Podemos modificar otras opciones en los ajustes del *input* como la velocidad del ratón u otros periféricos.

Todo esto se refleja en el *blueprint* del personaje, el cual tiene la forma de la figura de abajo. Para conseguir un mejor movimiento de la cámara, entre el jugador y la cámara, tenemos que añadir un *SpringArm,* que simula el efecto de un muelle entre el jugador y la "cámara" que contiene.

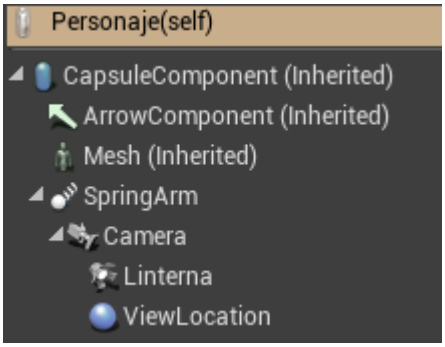

Figura 65. Estructura del blueprint del personaje

Juntando todo esto, obtenemos un personaje en primera persona básico con el que podemos empezar a añadirle más mecánicas.

5.3.3.2. Interacción

Cuando nos acercamos a una zona en la que podemos realizar alguna acción, esta se muestra con un símbolo de exclamación.

Para conseguir esto, debemos crear una función que añada un *widget* a la pantalla y otra para eliminarlo, y usar estas funciones cada vez que el personaje colisione con un *trigger* de acción.

Figura 66. Trigger de la puerta del dormitorio 1

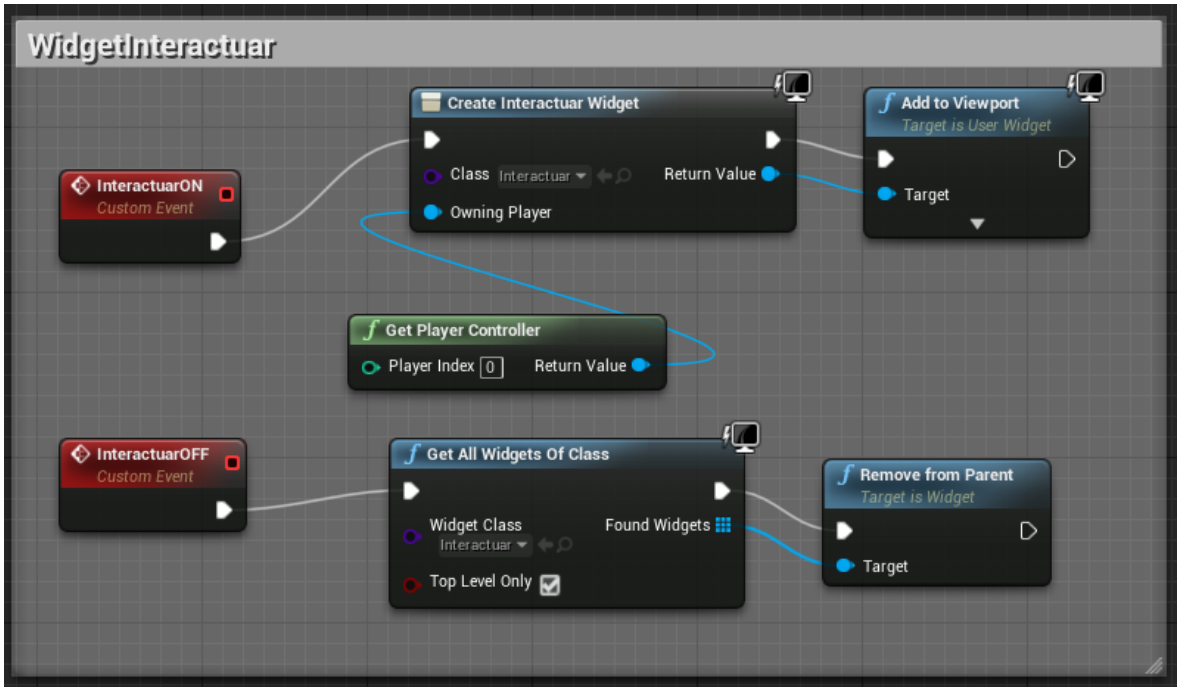

Figura 67. Blueprint del widget interactuar

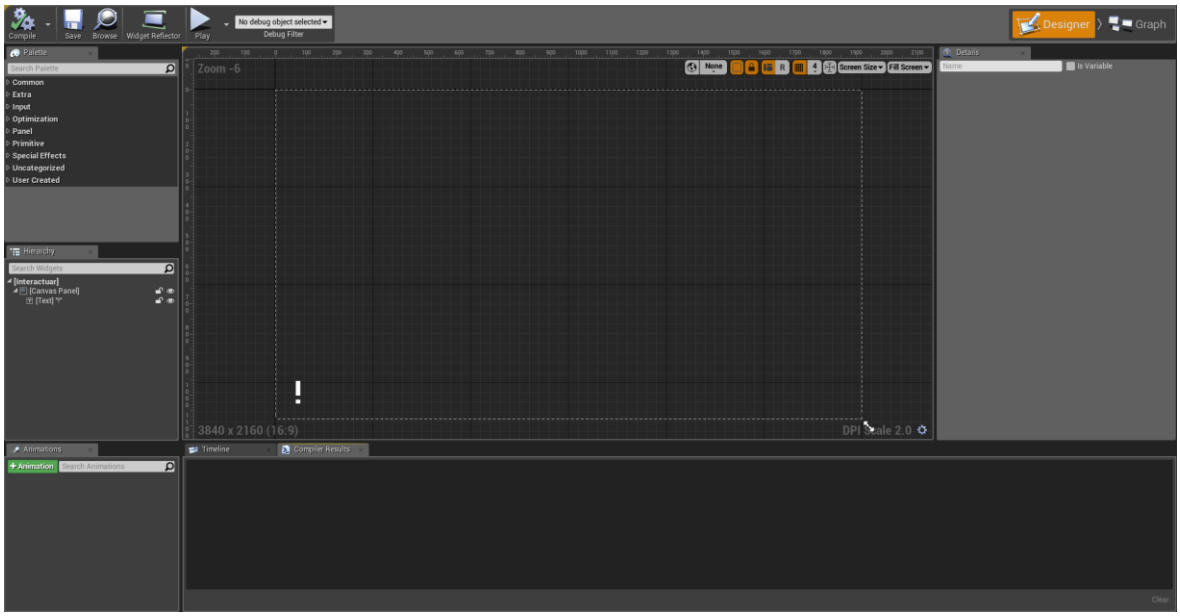

Figura 68. Widget interactuar

Esto simplemente es para remarcar que se puede hacer alguna acción. Tenemos diferentes acciones que podemos realizar en el nivel, como abrir puertas o accionar interruptores de la luz.

Todas estas acciones comparten el mismo comienzo, primero comprobamos si el jugador se encuentra en el *trigger* de acción y luego si ha pulsado el botón de interacción se realiza la acción o no.

Esto lo podemos observar en el siguiente *blueprint*:

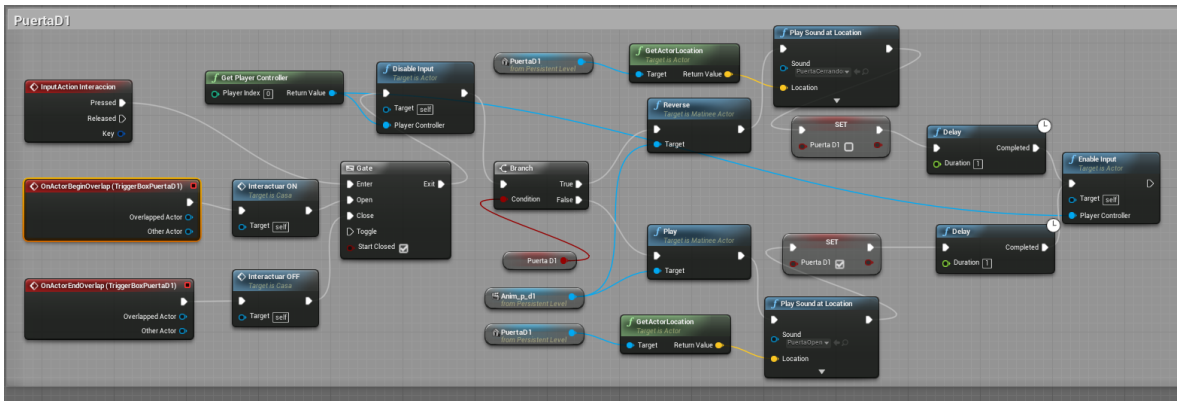

Figura 69. Blueprint de la puerta del dormitorio 1

En este caso es la acción de abrir la puerta, si la puerta estaba cerrada se realiza la animación de abrir la puerta, y si estaba abierta se cierra. Para no alargar innecesariamente este apartado, todas las acciones comparten de manera muy similar este comportamiento, variando la animación a realizar o cambiando la visibilidad de la luz, en el caso de los interruptores.

Antes de pasar al siguiente apartado, vamos a ver como se realiza una animación. Las animaciones en *Unreal Engine* se pueden realizar de diferentes formas, pero en este caso he decidido usar *Matinees*.

Un *matinee* es un objeto que nos permite definir diferentes posiciones de otro objeto. Por ejemplo, para abrir una puerta, tenemos que decirle al *matinee* que la puerta se encuentra en una posición de inicio y se rotará a una posición final en un tiempo determinado. El movimiento intermedio se interpola automáticamente.

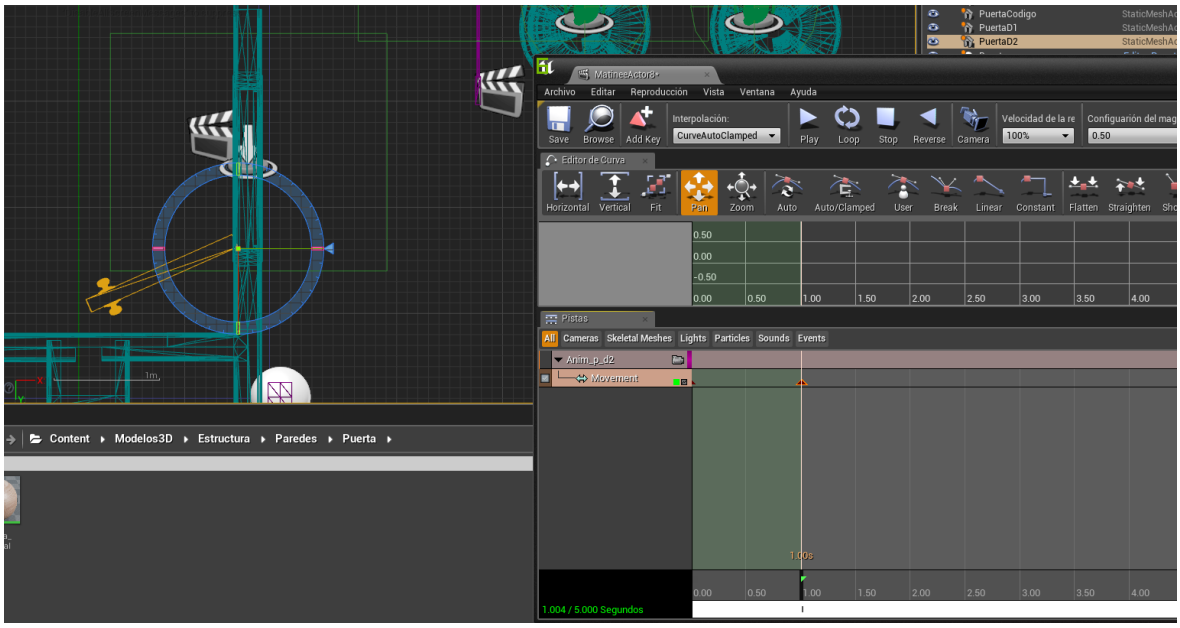

Figura 70. Matinee de la animación de una puerta

Para cada interacción del juego, se ha definido un *matinee* asociado al objeto en cuestión, reproduciéndose cuando se realiza la acción de interactuar.

Para la acción de interacción también debemos asociar una serie de *inputs*, al igual que hicimos con el movimiento y la cámara.

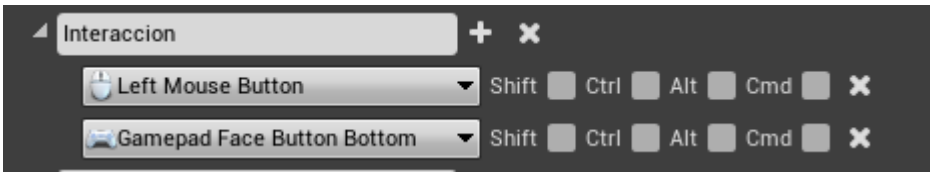

Figura 71. Mapeado de teclas de la acción interacción

5.3.3.3. Inspección

La inspección de objetos se lleva a cabo en el *blueprint* del jugador. Para ello tenemos que crear una función en la que, mediante un trazado de rayo desde el jugador a una cierta distancia, seamos capaces de saber si hay un objeto inspeccionable o no.

En el caso de que lo haya, tenemos que acercarlo a un componente del jugador que se encuentra delante de la cámara, desactivar el movimiento del jugador y trasladar el movimiento del ratón al movimiento del propio objeto. Una vez terminada la inspección, el objeto se vuelve a dejar en su posición original y se restablece el movimiento normal del jugador.

Todo esto se hace con varias funciones y comprobaciones para evitar que el jugador pueda moverse mientras se encuentra inspeccionando.

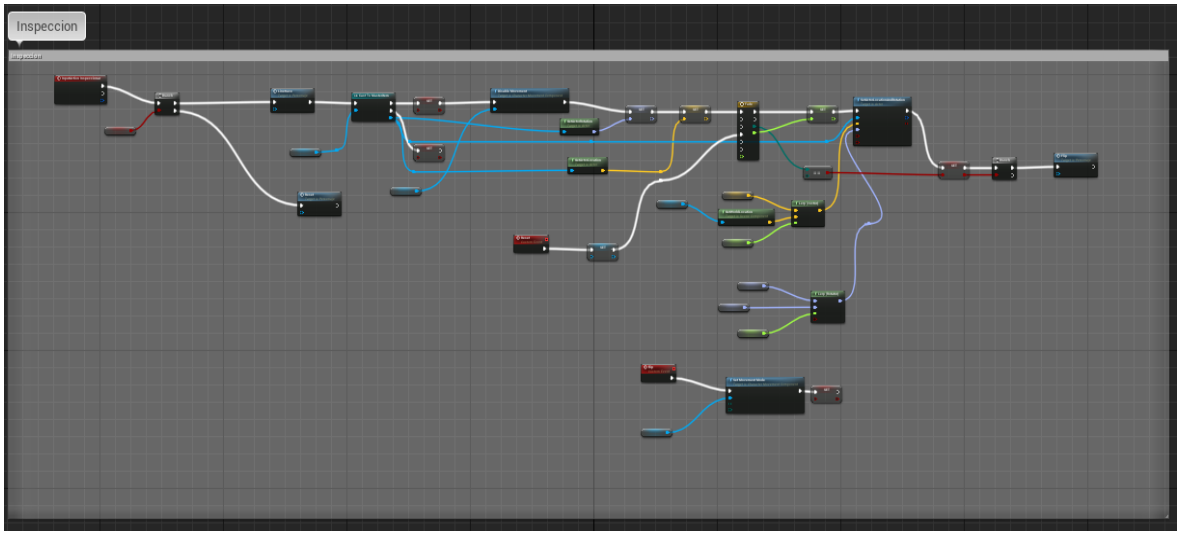

Figura 72. Función principal que controla la inspección de objetos

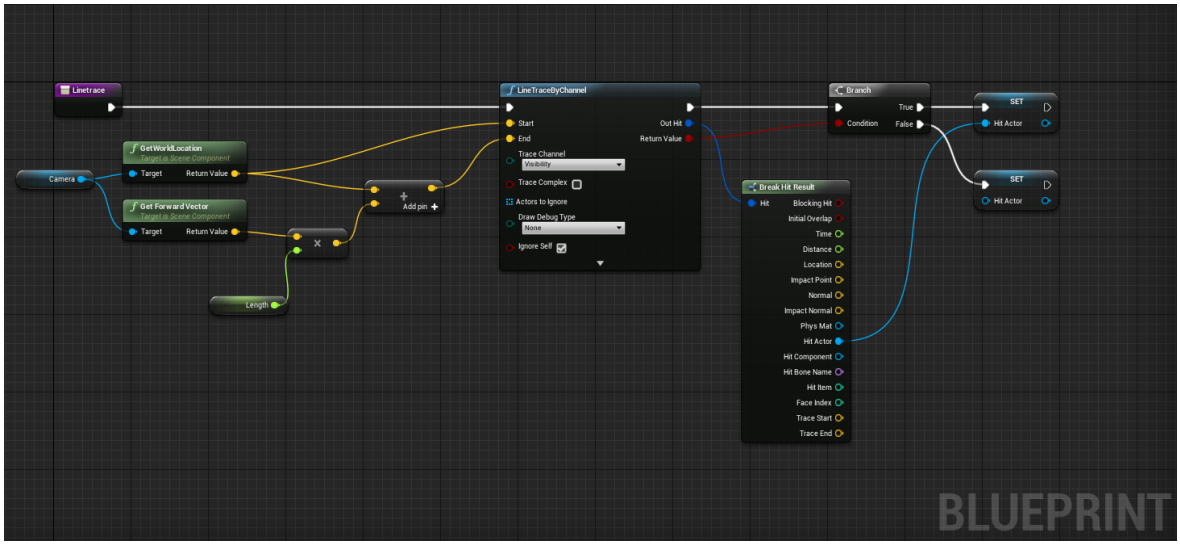

Figura 73. Función que se encarga del raytracing

Todos los objetos inspeccionables se tienen que crear uno a uno, ya que se componen de una colisión y una malla, esta ultima la vamos cambiando según el objeto que queramos.

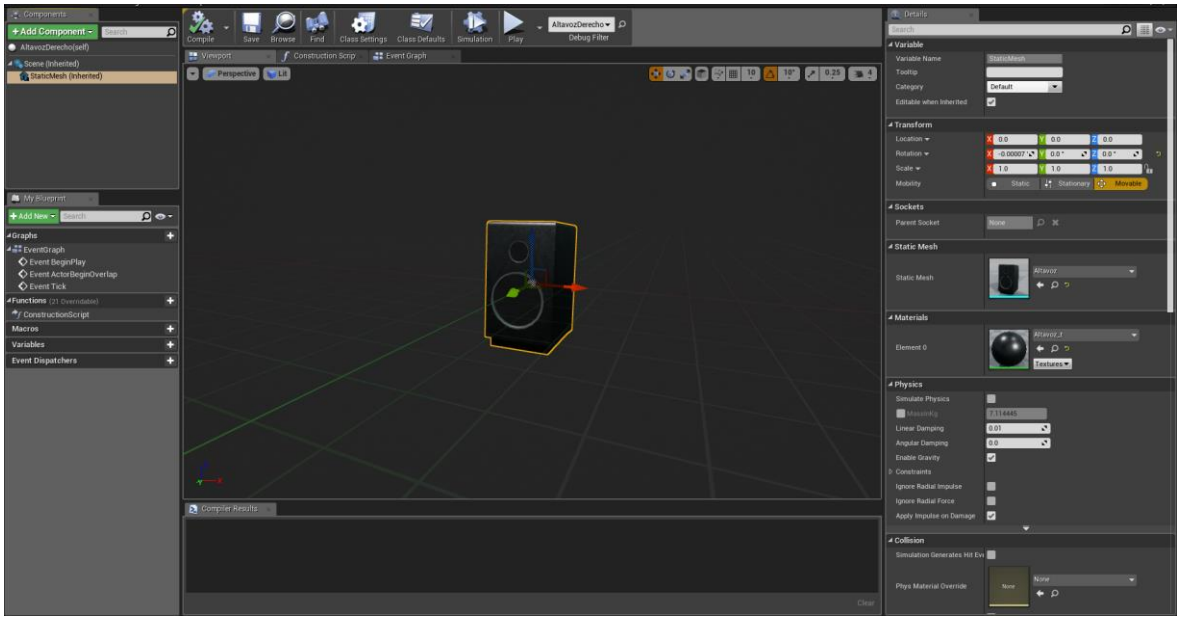

Figura 74. Objeto inspeccionable

Contamos con, aproximadamente, 90 objetos inspeccionables repartidos por todo el nivel.

Para la acción de inspección también debemos asociar una serie de *inputs*, al igual que hicimos con la interacción.

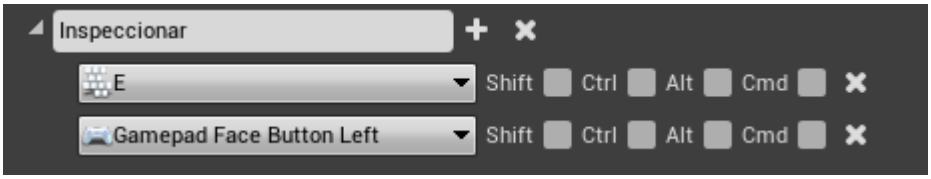

Figura 75. Mapeo de teclas para la acción inspeccionar

5.3.3.4. Enfoque

El enfoque es una mecánica muy sencilla, ya que solo tenemos que reducir o ampliar el campo de visión.

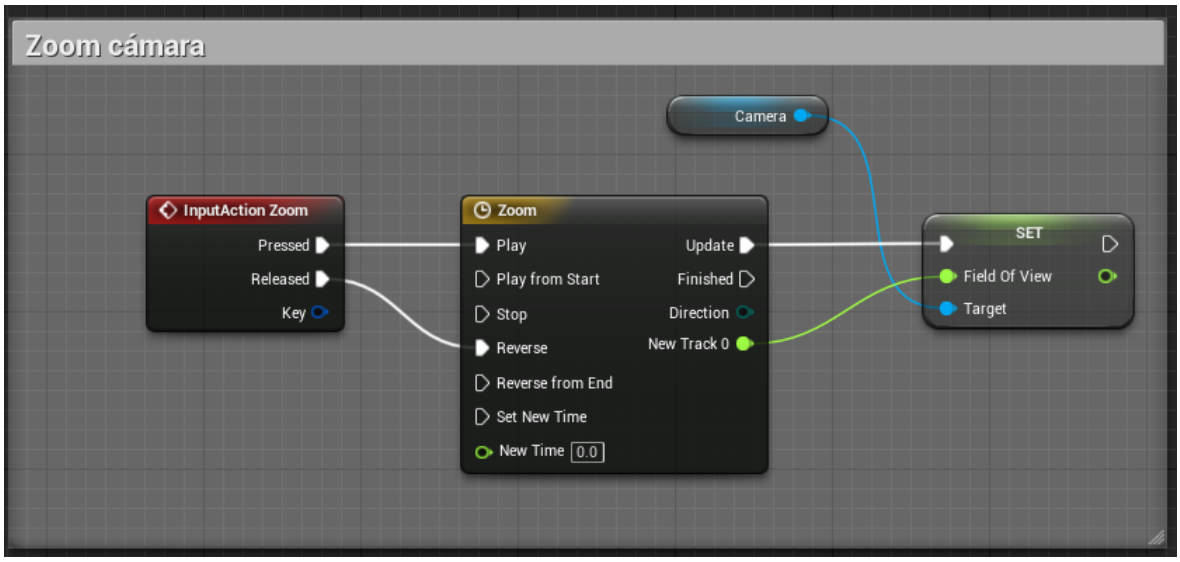

Figura 76. Blueprint de la función enfocar

Funciona de manera similar a un *matinee*, cuando realizamos la acción de enfocar, el *timeline* varía entre 90 y 60, lo que hace que se vea de más a menos.

El timeline funciona al presionar el botón y al soltar el botón, de forma que al pulsarlo pasa de 90 a 60 gradualmente el campo de visión de la cámara, y al soltarlo pasa de 60 a 90 nuevamente de manera gradual.

Para la acción de enfocar también debemos asociar una serie de inputs, al igual que hicimos con la interacción.

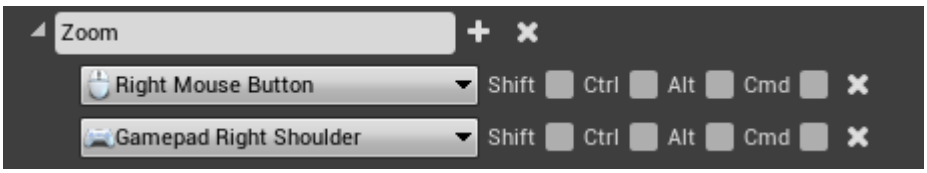

Figura 77. Mapeado de la acción enfocar

5.3.3.5. Linterna

La linterna también es una mecánica sencilla, ya que solo debemos conmutar la visibilidad de la luz que contiene el personaje.

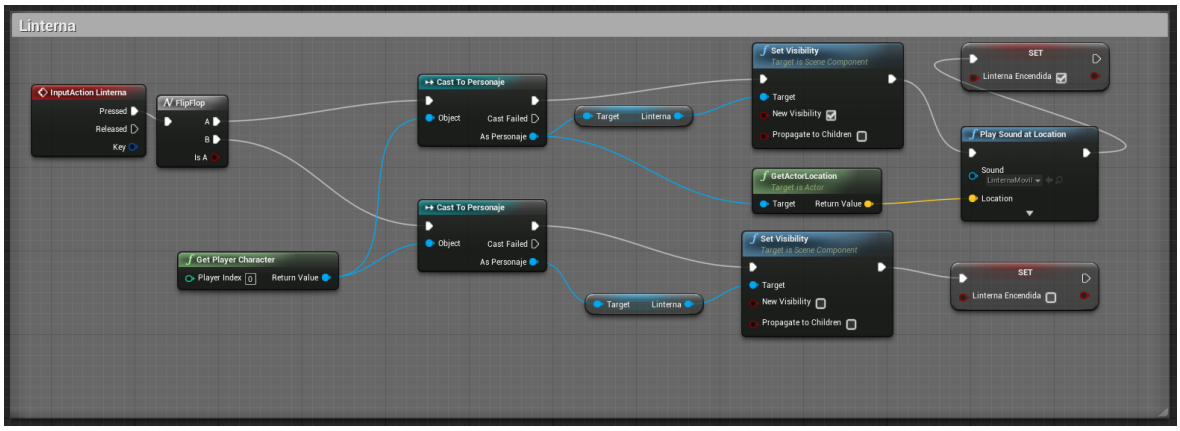

Figura 78. Blueprint de la función de la linterna

Para la linterna también debemos asociar una serie de inputs, al igual que hicimos con el enfoque.

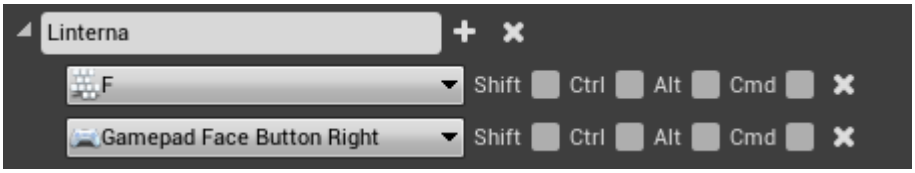

Figura 79. Mapeado de la acción linterna

5.3.3.6. Piezas Secretas

Las piezas secretas se comportan de manera diferente al resto de objetos interactivos, ya que resaltan a la vista del jugador y simplemente se cogen y se añaden al inventario.

Para conseguir esto debemos crear un actor, para poder manejar diferentes propiedades de las piezas y poder etiquetarlas.

Con el uso del trazado de rayo anteriormente comentado, podemos saber que objeto está mirando el jugador. Con esto y con la etiqueta que le hemos puesto podemos aislar este tipo de objeto del resto de la escena.

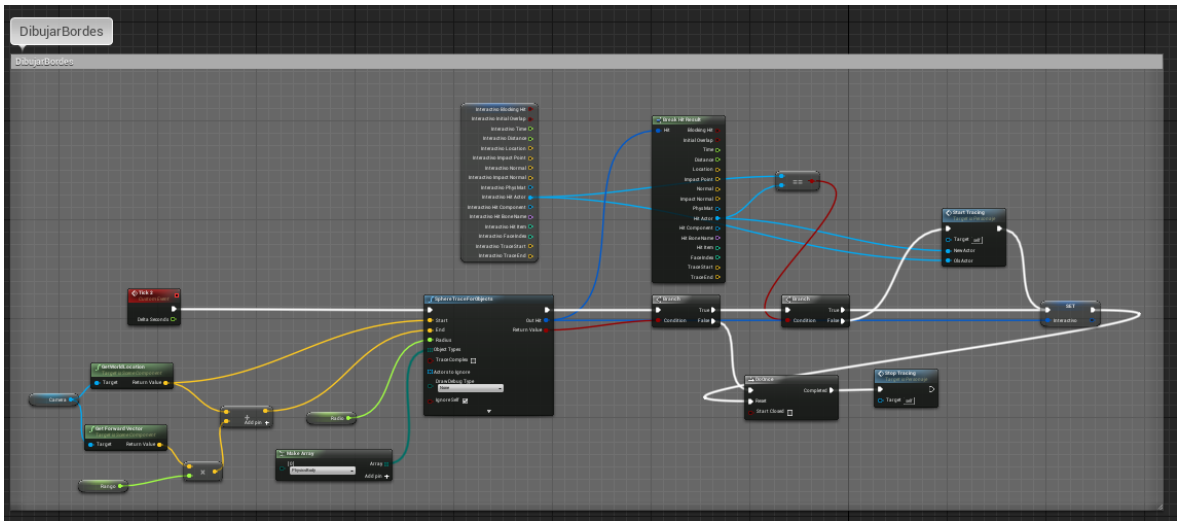

Figura 80. Blueprint principal para dibujar bordes

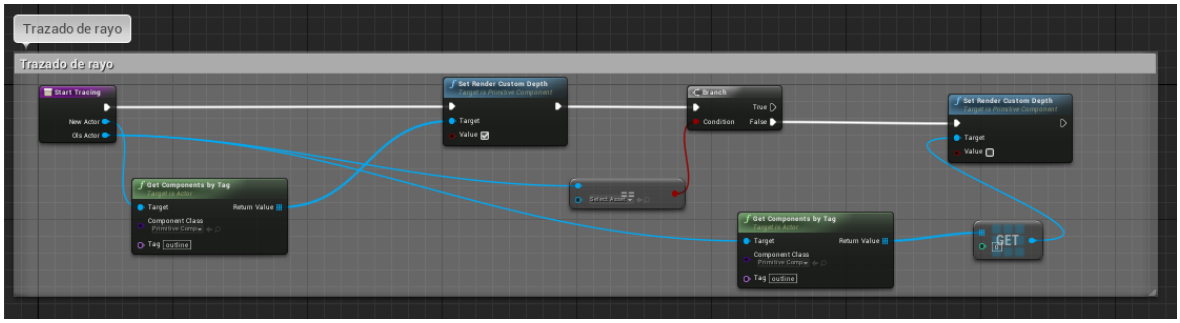

Figura 81. Comprobación del objeto al que mira el jugador

Para cogerlas es muy parecido a la acción de interactuar, cerca de ellas nos aparece la exclamación y al interactuar desaparecen de la escena y aparecen en el cuadro de la puerta secreta.

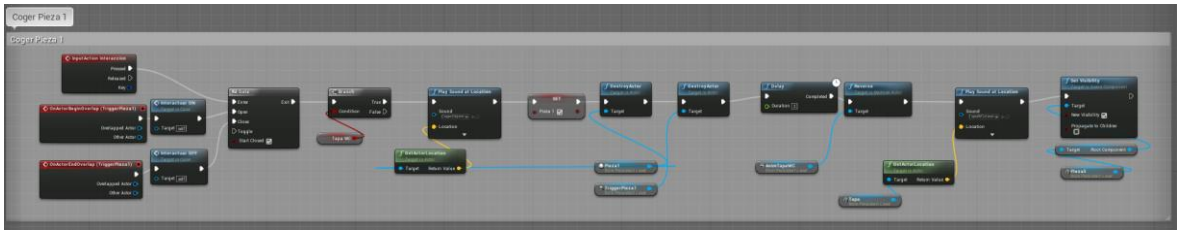

Figura 82. Función para coger una pieza y dibujarla en el cuadro de la puerta secreta

Actualmente se encuentran en una situación determinada. En un futuro se puede desarrollar una función para que cada vez que se juegue o para diferentes usuarios, no se encuentren en el mismo sitio.

5.3.3.7. Eventos

Hay diferentes eventos repartidos por la casa. Estos eventos simulan diferentes situaciones y acciones sobrenaturales. Por ejemplo, cuando entras al aseo y te acercas a espejo se enciende la luz y se cierra la puerta del aseo.

Otro ejemplo es que cuando cogemos la pieza de la cesta de ropa sucia, se apaga la luz y no se puede encender ninguna luz de la casa durante unos minutos, obligando al uso de la linterna.

Todo esto se consigue con elementos muy parecidos a los anteriores, como el uso de *triggers* o temporizadores, y mediante el uso de variables para tenerlo todo bajo control.

Con esto se contrarresta la falta de IA, ya que por la naturaleza del proyecto no se podía realizar en el tiempo dado, y se puede asustar al jugador de múltiples maneras.

5.3.4. Multimedia

En este apartado vamos a ver uno de los aspectos que no puede faltar en ningún videojuego, como es el apartado multimedia o comúnmente conocido como los sonidos y efectos dentro de un videojuego.

Se va a analizar tanto el sonido utilizado en el juego, como los vídeos y las técnicas de reproducción de estos.

5.3.4.1. Sonido

Para conseguir una buena inmersión del jugador, un juego debe tener unos buenos sonidos y en los sitios clave, para transmitir cosas que de otras maneras no podríamos.

En *Unreal Engine* podemos hacer infinidad de cosas con sonidos, aunque en nuestro caso se ha optado por sonidos previamente editados y puestos directamente en el juego, activándose cuando los necesitamos.

Vamos a empezar por lo más simple, como es por ejemplo el sonido de una puerta.

Al importar un sonido a *Unreal Engine*, se nos genera un *sound wave*, el tipo de archivo predefinido para el uso de sonidos.

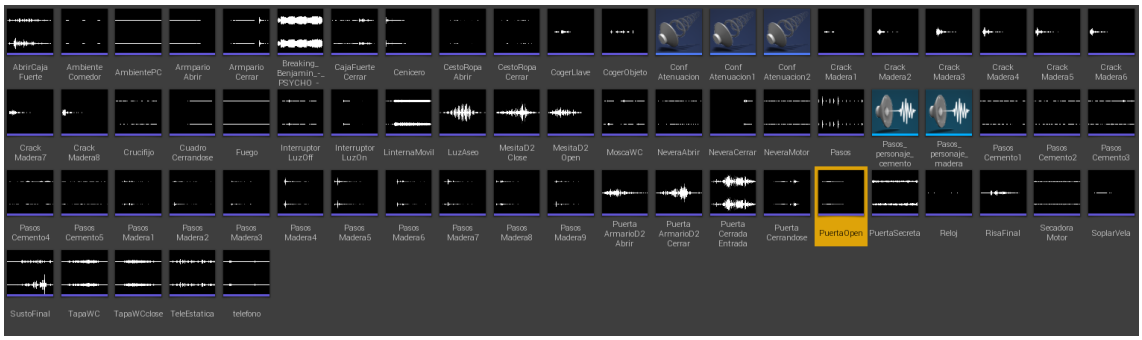

Figura 83. Algunos sound waves utilizados para el juego

Entre los diferentes sonidos seleccionamos el de la puerta y podemos ver las acciones básicas que podemos modificar, como el volumen, si queremos que el sonido se repita en bucle, entre otras muchas.

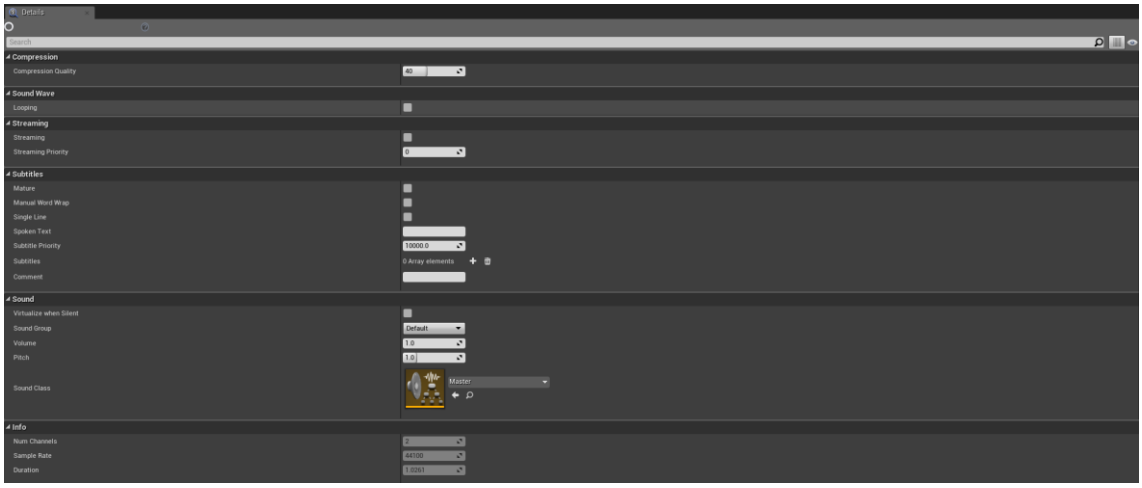

Figura 84. Propiedades de las sound waves

También para transmitir más realismo a la escena, podemos crear un *sound attenuation*, que consiste en dotar al sonido de las propiedades que tenemos en el mundo real, como son la atenuación que consigue que el sonido sea más fuerte en el origen y se desvanezca en la distancia.

Aquí lo más importante es que el tipo de atenuación sea logarítmica y ajustar los radios según nos convenga.

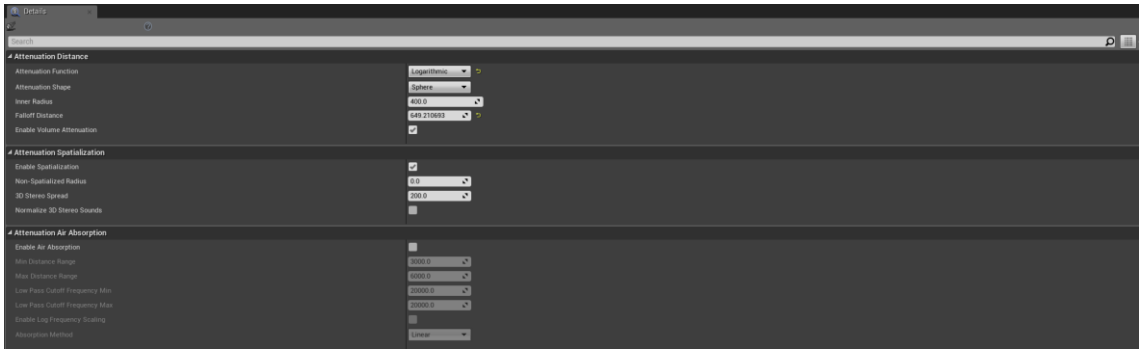

Figura 85. Propiedades de la atenuación del sonido

Una vez realizados estos ajustes, pasamos a utilizar el sonido. La forma más simple es cada vez que realicemos determinada acción, llamar al sonido correspondiente, aplicándole el *sound attenuation*.

En el caso de abrir la puerta, una vez realizada la llamada a la animación se realiza inmediatamente la llamada al sonido, en la localización de la puerta.

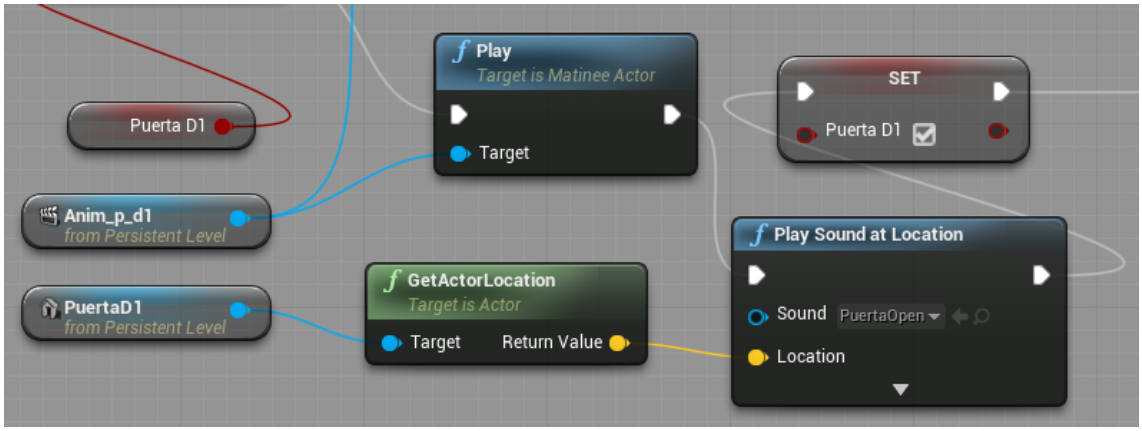

Figura 86. Blueprint usado para reproducir sonidos

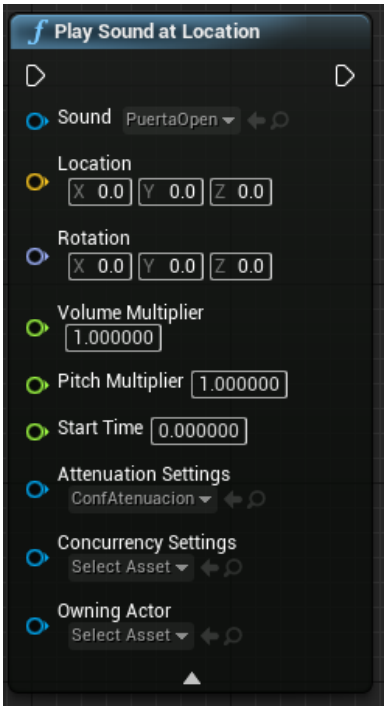

Figura 87. Blueprint para ejecutar un cierto sonido y diferentes propiedades

La mayoría de los sonidos se han realizado de la misma forma, excepto los pasos del personaje, los cuales se han tenido que preparar ligeramente diferentes.

Para los pasos se ha generado el *sound cue*, el cual es parecido al *sound wave*, pero nos permite contener varios *sound wave* dentro del él. Para dar dinamismo a los pasos, se han generado diferentes tipos, así como unos crujidos en la madera. Todos los sonidos pasan por un *random* para generar variedad y acaban juntándose como un único sonido a la salida.

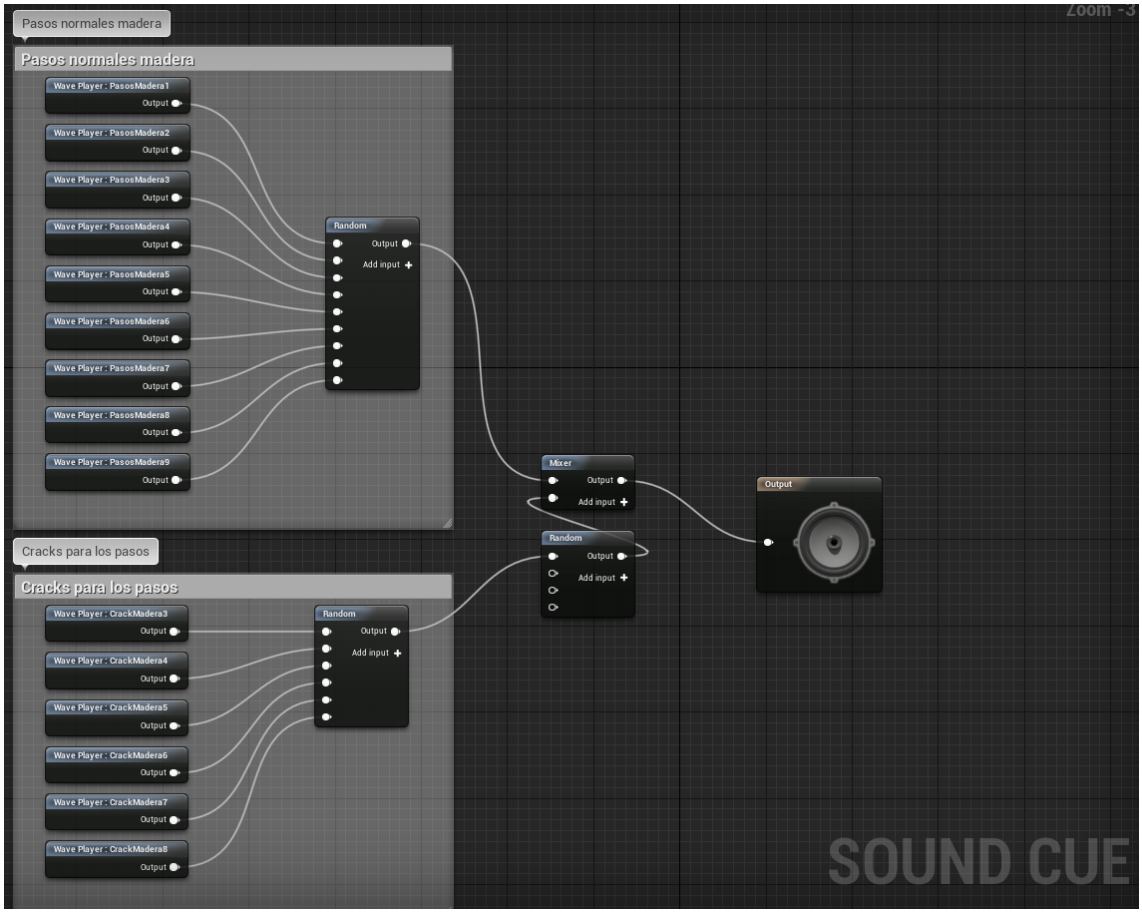

Figura 88. Pasos en superficie de madera

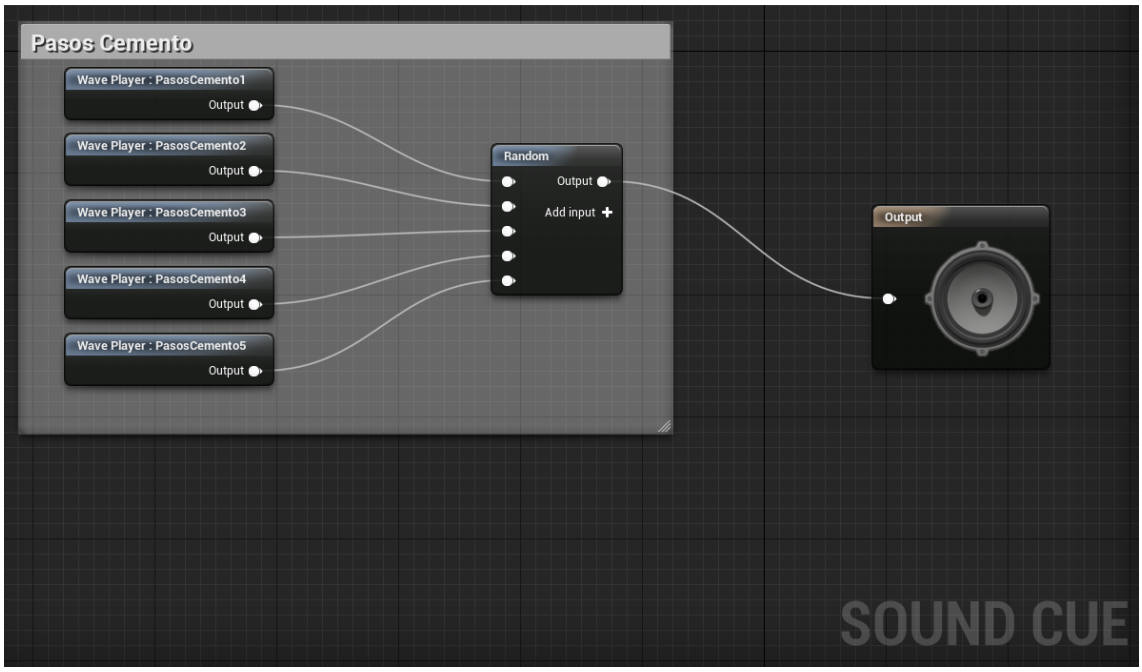

Figura 89. Pasos en superficie de cemento

Como se puede observar, tenemos pasos de madera y de cemento, esto es así porque vamos a distinguir la superficie en la que estamos. Para ello tenemos que saber cuándo el personaje está andando, así como la superficie en la que se encuentra.

Para saber cuándo el personaje está andando, solo tenemos que guardarnos una variable cada vez que de reciba un cambio en el movimiento.

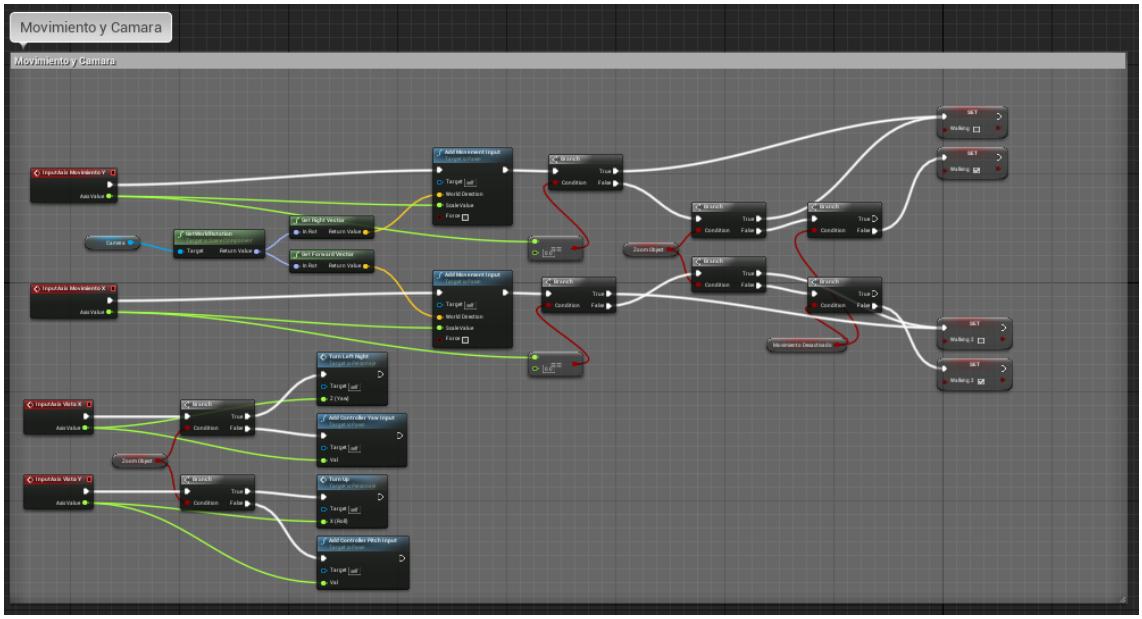

Figura 90. Comprobación de si el personaje se mueve o no

Para saber que superficie se encuentra debemos usar un rayo desde nuestra posición hasta el suelo, que al chocar podamos guardar en una variable el *physic material* que encuentre.

El *physic material* es un material físico que añadimos a otro material para decirle a *Unreal Engine* que ese material representa una superficie, previamente definida en los ajustes del proyecto.

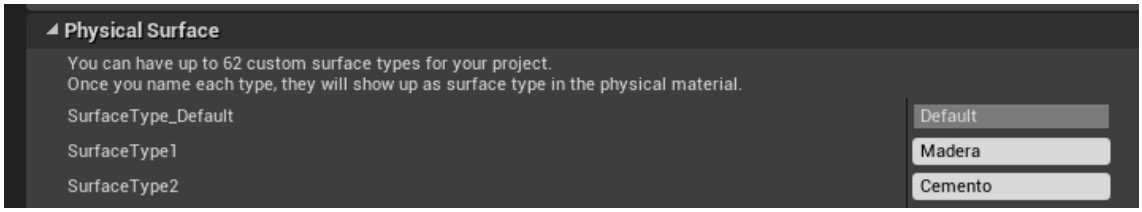

Figura 91. Declaración de los materiales físicos

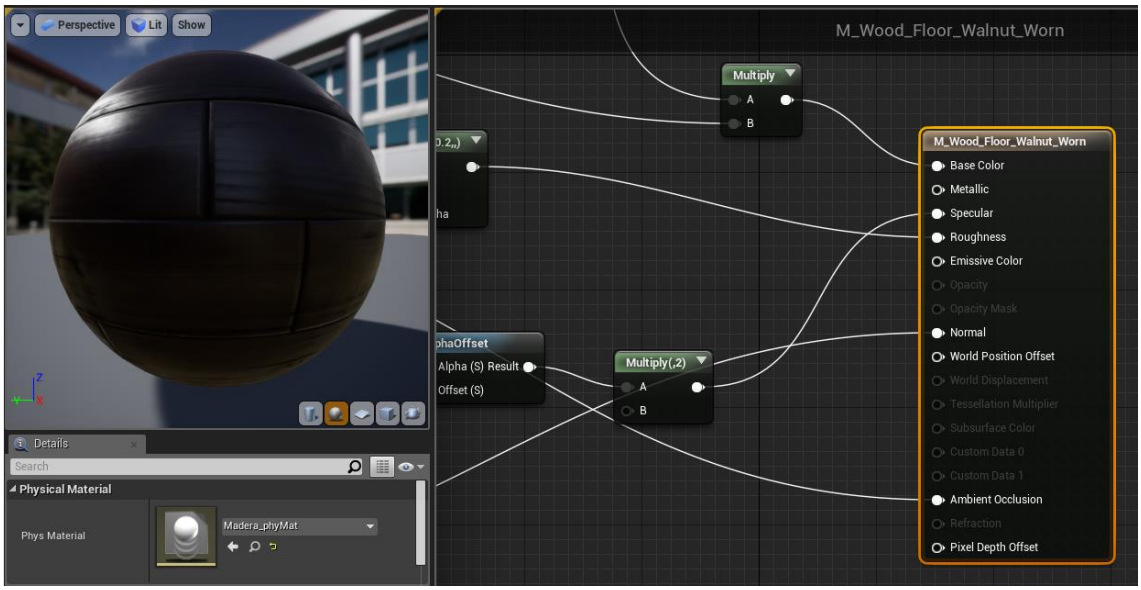

Figura 92. Asignación del material físico al suelo de madera

Una vez asignado el material físico a un material que queramos, ya podemos diferenciar superficies mediante el trazado de rayo hacia el suelo.

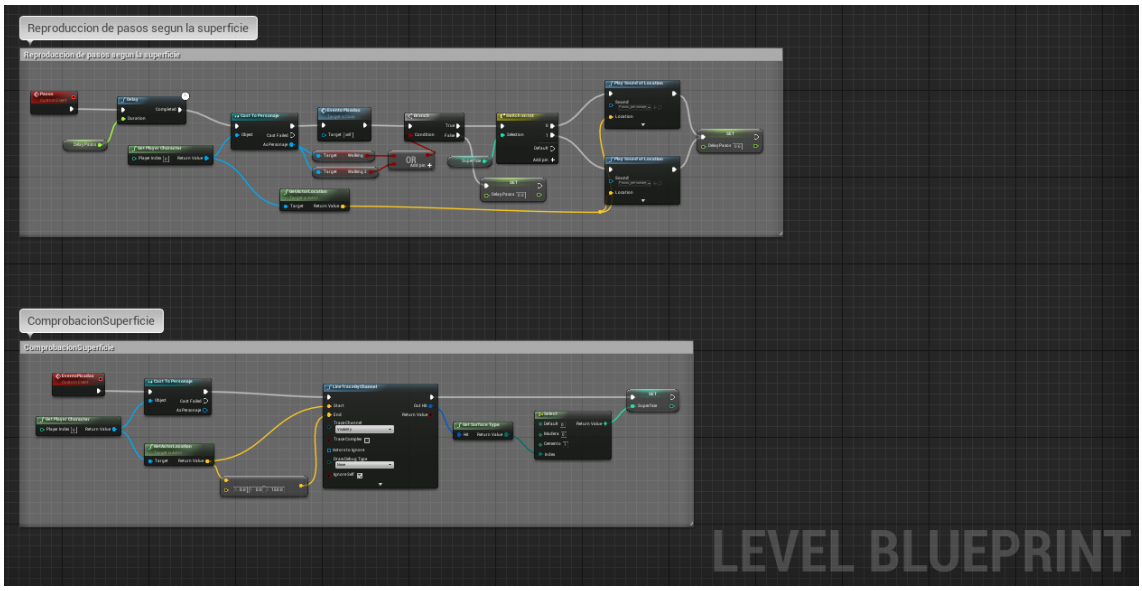

Figura 93. Blueprints para diferenciar entre superficies y reproducir el sonido deseado

Todo esto nos permite tener diferentes superficies y diferentes sonidos para cada una.

5.3.4.2. Video

En algunos objetos de la casa, como las pantallas del ordenador o la televisión del salón, se reproducen videos para asustar al jugador o darle alguna pista.

Para ello debemos crear el video mediante el programa de edición que prefiramos, en mi caso *Adobe after Effects* junto a *Adobe Premiere*, e importarlo a *Unreal Engine*.

Una vez hecho esto, tenemos que crear un mediaplayer, el cual es una especie de reproductor de video al que le podemos asociar diferentes videos, así como que se repitan o no.

Al crearlo, también nos genera una textura de video para poder aplicárselo a cualquier material.

Figura 94. De derecha a izquierda: video, mediaplayer, textura de video

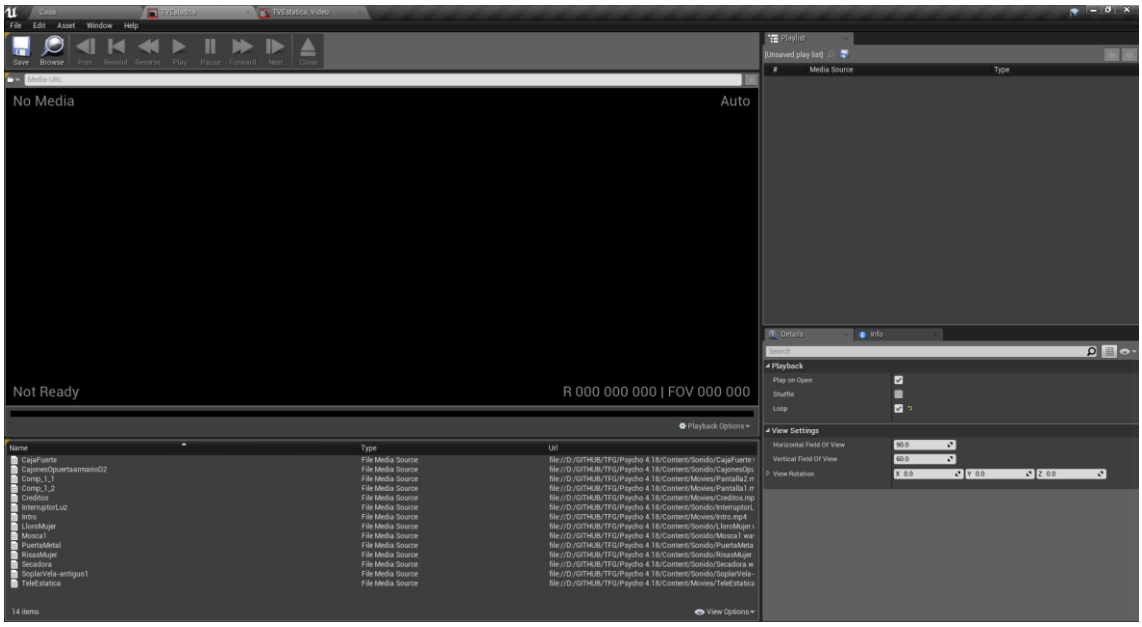

Figura 95. Apariencia del mediaplayer

Con todo lo anterior realizado, ya podemos aplicar la textura directamente a cualquier modelo y cuando activemos la reproducción, se verá el video en dicho material.

En nuestro caso, como anteriormente vimos en la sección de materiales, hemos creado un material exclusivo para las pantallas de televisión. Con este material conseguimos el efecto de que se puedan ver los pixeles desde cerca, al igual que ocurre con una pantalla real.

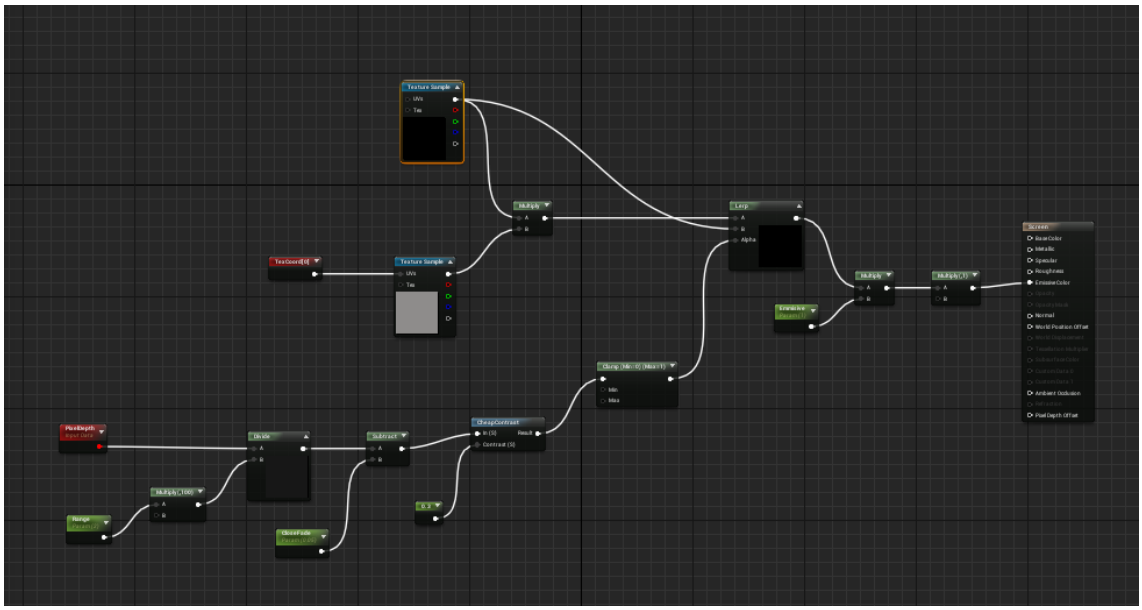

Figura 96. Material para pantallas

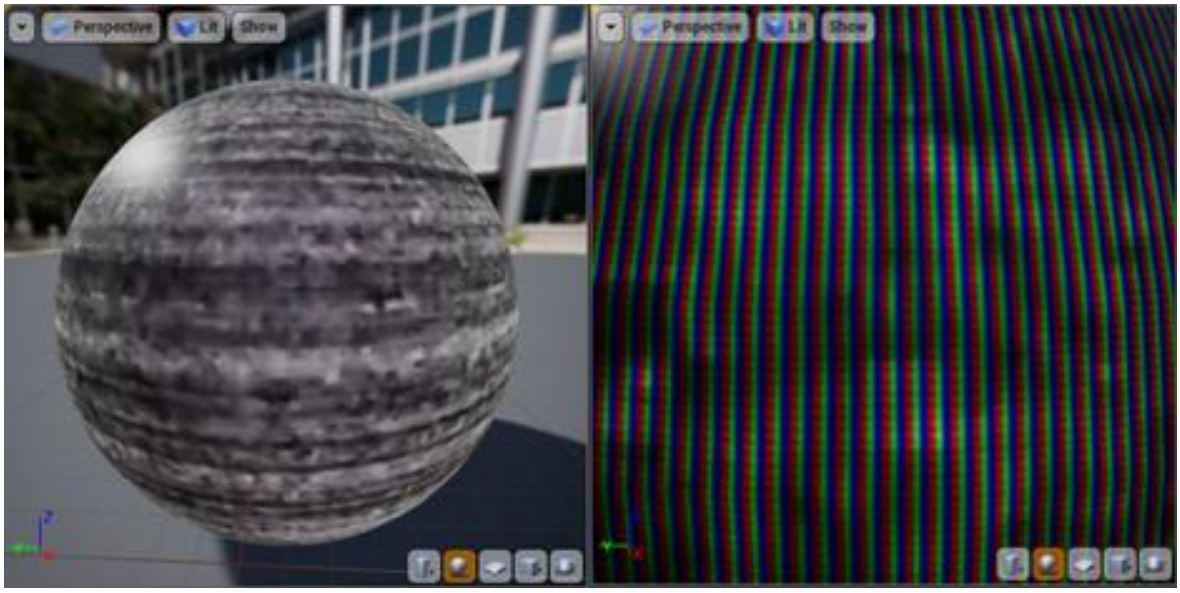

Figura 97. Vista del material para las pantallas

Para conseguir este efecto, necesitamos la imagen real de un pixel, formada por los tres colores RGB, y mediante una serie de nodos y operaciones conseguimos que la textura del video del mediaplayer interactúe con esa imagen, obteniendo el efecto de que cada pixel emite el color deseado.

También se le pueden aplicar otros efectos como la emisión de luz de la textura para dar mayor realismo, así como la profundidad visual de cada pixel.

5.3.5. Menús

Una parte muy importante de un videojuego son los menús. Es lo primero que el jugador verá y le puede dar una idea de cómo será el estilo visual del videojuego.

Para la elaboración de los menús se han utilizado los *widgets* de *Unreal Engine*, junto a diferentes imágenes y textos.

Los menús en ***Psycho*** son cuatro principales más dos cinemáticas de inicio y de créditos.

5.3.5.1. Menú Principal

Para el menú principal se ha optado por la simplicidad, resaltando el título del juego y 3 opciones: jugar, controles y salir del juego.

Figura 98. Imagen del menú principal

Cada botón tiene 3 estados: el estado por defecto, el estado cuando esta seleccionado, y el estado pulsado.

Figura 99. Botones menú

Figura 100. Widget del menú principal

5.3.5.2. Controles

En este menú se pueden consultar los controles del juego, tanto para teclado y ratón como para mando. Cuenta con una estética muy simple y minimalista que muestra todo de forma visual.

Únicamente cuenta con un botón para volver atrás.

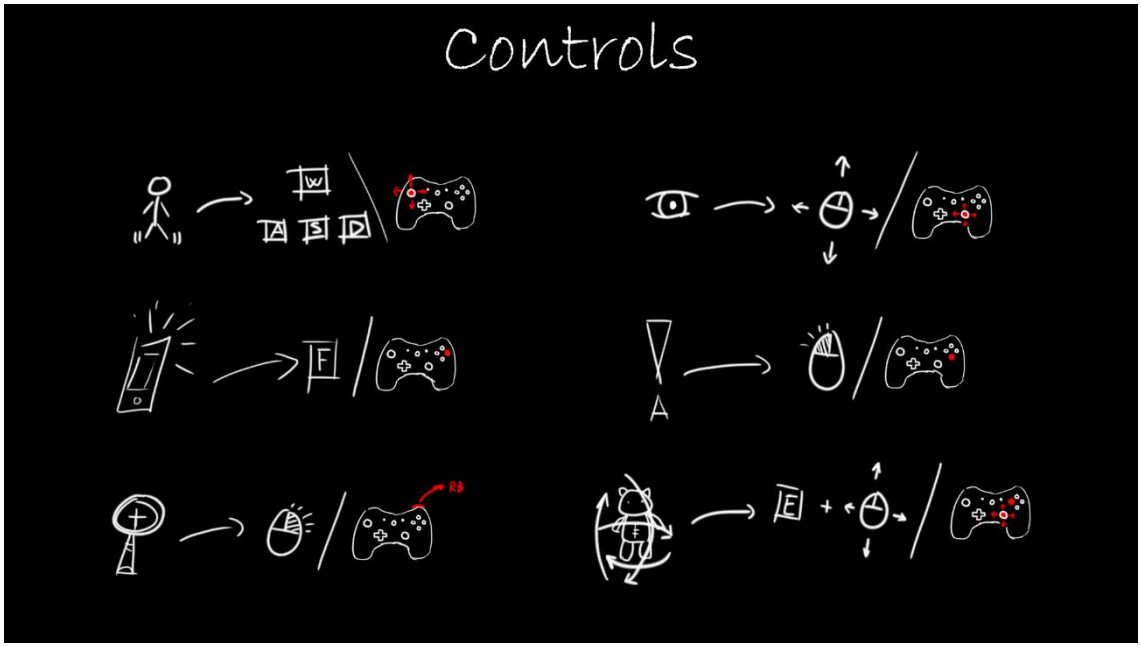

Figura 101. Imagen de los controles del juego

5.3.5.3. Pantalla de carga

La pantalla de carga es un recurso que utilizamos para ofrecer alguna información extra al jugador mientras carga el juego.

En este caso se ha optado por utilizar una imagen que nos indica que, para disfrutar de una mejor experiencia, se recomienda el uso de auriculares, ya que como comentamos anteriormente, el audio es en 3D.

Figura 102. Imagen de la pantalla de carga

5.3.5.4. Configuración

En este menú, el cual solo se puede ejecutar dentro del juego pulsando la tecla escape o el botón start del mando, se pueden ajustar diferentes opciones gráficas, como la sincronización vertical, la calidad de las texturas, etc.

Una vez realizados los ajustes necesarios, cuando volvamos al juego se aplicarán los cambios.

También desde este menú podemos volver al menú principal.

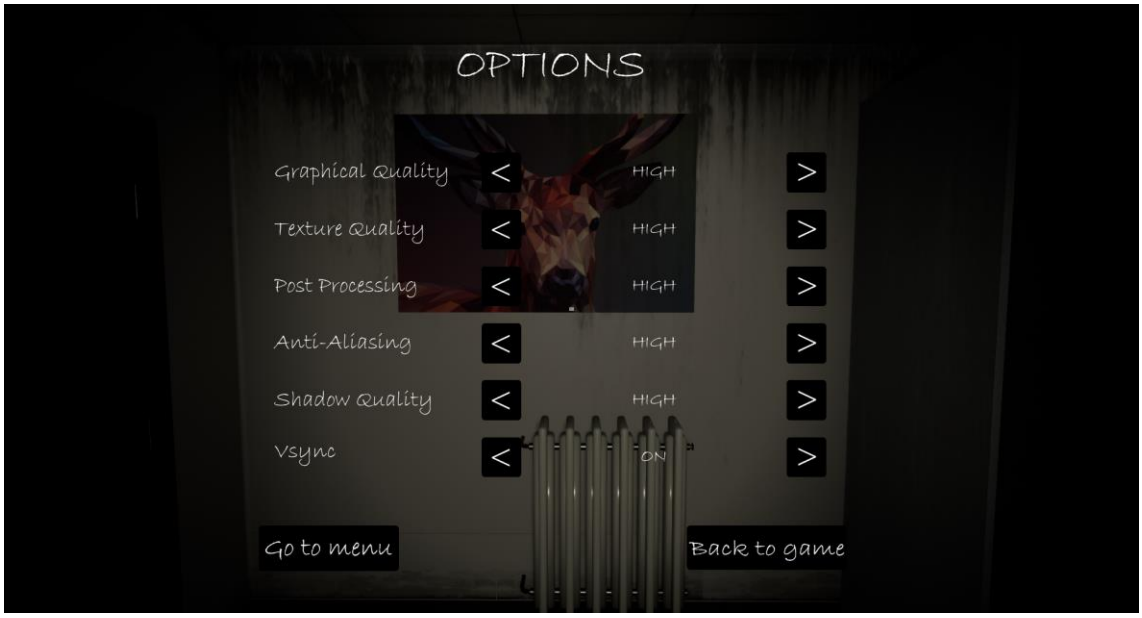

Figura 103. Imagen InGame del menú de opciones

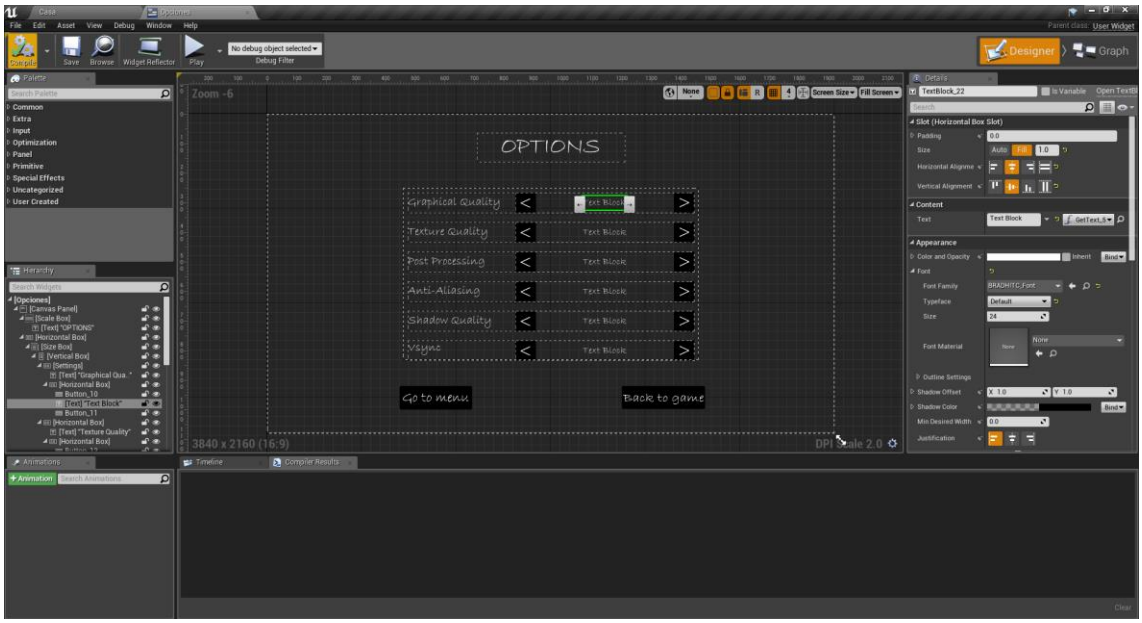

Figura 104. Widget del menú de opciones

Los datos de estos ajustes se guardan en el jugador, y principalmente son una serie de comandos que modifican la consola de comandos de *Unreal Engine*.

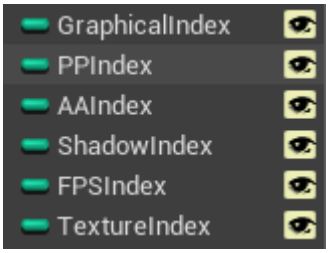

Figura 105. Variables públicas para la configuración

Usando esas variables obtenemos la configuración inicial por defecto, y la aplicamos al iniciarse el juego.

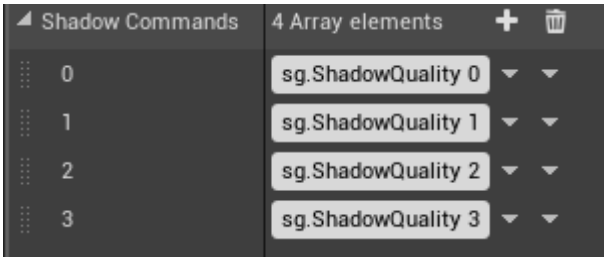

Figura 106. Lista de comandos para la calidad de las sombras

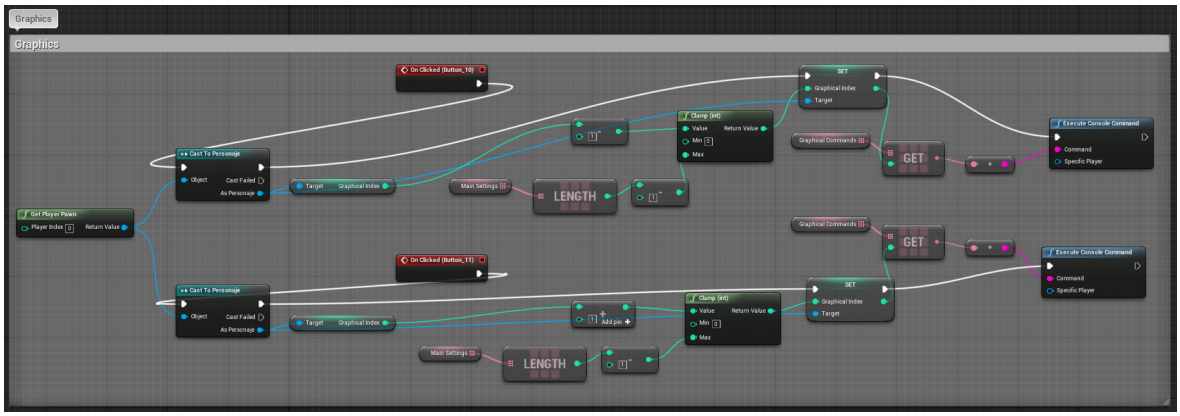

Figura 107. Blueprint para aplicar la configuración deseada

La forma en la que se introducen estos comandos es guardarlos en listas y ejecutando el *blueprint* para ejecutar comandos en la consola.

Sabemos qué comando corresponde a cada opción porque asociamos a cada configuración un número y texto, y posteriormente lo comparamos con la posición de la lista, obteniendo así el comando deseado.

De forma similar se utiliza este procedimiento para el resto de las opciones gráficas.

5.3.5.5. Cinemática inicial

Cuando se ejecuta el juego, antes de que aparezca el menú principal, se puede ver un video que nos muestra el logo de *Unreal Engine* y la calificación de edad.

Este video se ha realizado mediante *After Effects* y *Adobe Premiere* y se ha introducido en los ajustes del proyecto, para que se reproduzca al iniciar el juego.

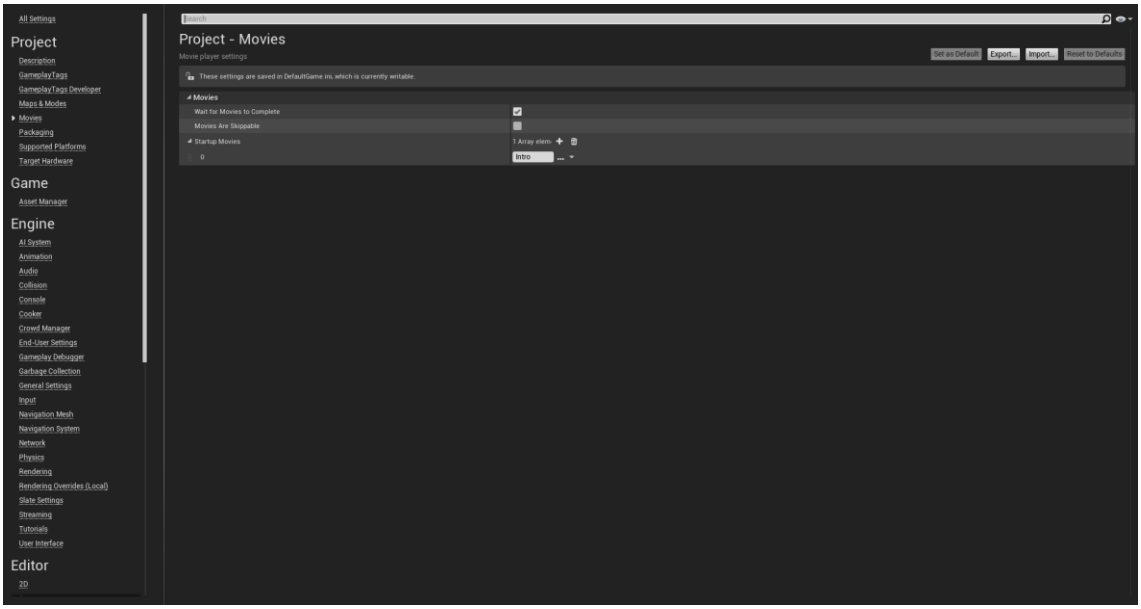

Figura 108. Declaración de la cinemática inicial en los ajustes del proyecto

Figura 109. Captura cinemática inicial

5.3.5.6. Créditos

Después de los dos finales se reproduce un video donde se muestran los créditos del juego.

Este video ha sido realizado de la misma forma que la cinemática inicial, mediante *After Effects* y *Adobe Premiere* y se ha ejecutado el video dentro del juego, mediante el uso de *widgets*.

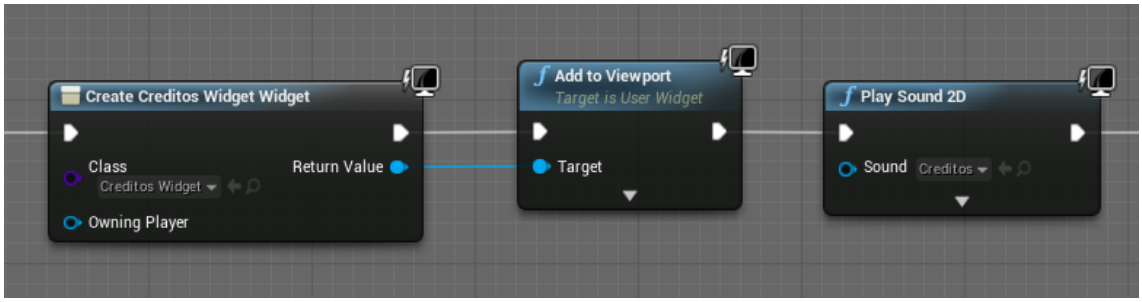

Figura 110. Blueprint para ejecutar los créditos

Figura 111. Captura de los créditos

5.3.6. Realidad Virtual

El género y estilo del juego favorecen el uso de la realidad virtual por lo que se han realizado pruebas e investigaciones sobre la inclusión de realidad virtual en ***Psycho*** y en un futuro, fuera de este trabajo, se realizará una adaptación a realidad virtual completa para poder disfrutar de una mejor experiencia inmersiva.

6. Conclusiones

Tras varios meses de trabajo y esfuerzo, se ha conseguido un producto acabado y completamente jugable. Podría considerarse un juego completo, ya que la duración de este puede variar según el público que lo juegue, entre 10 y 30 minutos e incluso más, dependiendo del nivel de detalle que el jugador quiera observar.

En cuanto a la experiencia adquirida con este proyecto ha sido impresionante. He mejorado mi nivel de modelado 3D, adquiriendo nuevas técnicas que desconocía y analizando los objetos de una forma diferente, pensando en la optimización para videojuegos.

También he mejorado la calidad de la texturización y descubierto diferentes programas que son verdaderamente útiles en la creación de modelos para un videojuego.

Este trabajo ha supuesto un desafío, ya que he asumido todos los roles de un videojuego, desde el *game design* o diseño de videojuego hasta la publicación de este, pasando por los roles de modelador, texturizador, diseñador de niveles, técnico de sonido y programador.

Cabe destacar que se ha enfocado más al modelado y texturizado, pero que no se ha dejado ningún detalle al azar y se ha querido realizar un producto desde cero.

Como bien me han enseñado en la carrera de Ingeniería Multimedia a lo largo de estos años, lo más importante es querer tu producto, verlo crecer, darle valor como producto que es, y saber venderlo bien al resto de la gente, manteniéndolos al día de los avances y logros que se consiguen, mediante el uso de redes sociales.

Esto lo he llevado a cabo ya que se han ido realizando publicaciones del avance del juego, e incluso he conocido a personas interesadas por el juego final. También he realizado un tráiler del juego y subido a una plataforma para que la gente lo pueda disfrutar y así, espero, darme a conocer.

Por último, aunque considere el juego como un juego acabado, sí que me gustaría realizar algunas mejoras en un futuro, como la adaptación a realidad virtual o la inclusión de una IA que merodee por la casa, o incluso ampliarlo a una casa más grande o vecindario. Gracias al uso de un entorno modular, esto no supondría ningún problema y se podría realizar de forma rápida.

7. Bibliografía

[1]. Definición.de: https://definicion.de/videojuego/

[2]. Wikipedia.org: https://es.wikipedia.org/wiki/Horror_de_supervivencia

[3]. Wikia.com: http://es.emily-wants-to-play.wikia.com/wiki/Wikia_Emily_Wants_To_Play

[4]. Wikipedia.org: https://en.wikipedia.org/wiki/Emily_Wants_to_Play

[5]. Wikia.com: http://markiplier.wikia.com/wiki/Play_With_Me

[6]. Wikipedia.org: https://en.wikipedia.org/wiki/P.T._(video_game)

[7]. Wikipedia.org: https://en.wikipedia.org/wiki/Layers_of_Fear

[8]. Wikipedia.org: https://en.wikipedia.org/wiki/Resident_Evil_7:_Biohazard

[9]. Wikipedia.org: https://es.wikipedia.org/wiki/Kanban_(desarrollo)

[10]. Wikipedia.org: https://es.wikipedia.org/wiki/Unity_(motor_de_juego)

[11]. Wikipedia.org: https://es.wikipedia.org/wiki/Unreal_Engine

[12]. Wikipedia.org: https://es.wikipedia.org/wiki/Blender

[13]. Wikipedia.org: https://es.wikipedia.org/wiki/Autodesk_Maya

[14]. Wikipedia.org: https://es.wikipedia.org/wiki/Autodesk_3ds_Max

[15]. Wikipedia.org: https://es.wikipedia.org/wiki/Mudbox

[16]. Wikipedia.org: https://es.wikipedia.org/wiki/Adobe_Photoshop

[17]. Audacityteam.org: https://www.audacityteam.org/

[18]. Freesound.org: https://freesound.org/

[19]. Eldocumentalistaudiovisual.com: https://eldocumentalistaudiovisual.com/2015/02/06/documentacion-en-videojuegos-documento-de-diseno-gdd/

[20]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted

[21]. Docs.unrealengine.com: https://docs.unrealengine.com/en-us/Engine/Blueprints

[22]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types

[23]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/LevelBlueprint

[24]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/Types/ClassBlueprint

[25]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/Basics

[26]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightTypes

[27]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightMobility

[28]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/IntroductionToMaterials

[29]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/MaterialInputs

[30]. Docs.unrealengine.com: https://docs.unrealengine.com/en-US/Engine/Content/Types/Tetures

8. Anexo: Arte

En este apartado vamos a ver diferentes capturas del juego final.

Figura 112. Captura InGame de Psycho1

Figura 113. Captura InGame de Psycho 2

Figura 114. Captura InGame de Psycho 3

Figura 115. Captura InGame de Psycho 4

Figura 116. Captura InGame de Psycho 5

Figura 117. Captura InGame de Psycho 6

Figura 118. Captura InGame de Psycho 7

Figura 119. Captura InGame de Psycho 8

Figura 120. Captura InGame de Psycho 9

Figura 121. Captura InGame de Psycho 10

Figura 122. Captura InGame de Psycho 11

Figura 123. Captura InGame de Psycho 12

Figura 124. Captura InGame de Psycho 13

Figura 125. Captura InGame de Psycho 14

Figura 126. Captura InGame de Psycho 15

Figura 127. Captura InGame de Psycho 16

Figura 128. Captura InGame de Psycho 17

Figura 129. Captura InGame de Psycho 18

Figura 130. Captura InGame de Psycho 19

Figura 131. Captura InGame de Psycho 20

Figura 132. Captura InGame de Psycho 21

Figura 133. Captura InGame de Psycho 22

Figura 134. Captura InGame de Psycho 23

Figura 135. Captura aérea del nivel